2001-09

Integrated Development Environment (IDE) for the construction of a Federation Interoperability Object Model (FIOM)

Young, Paul E.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

INTEGRATED DEVELOPMENT ENVIRONMENT (IDE) FOR THE CONSTRUCTION OF A FEDERATION INTEROPERABILITY OBJECT MODEL (FIOM)

by

Brent P. Christie
Paul E. Young

September 2001

Thesis Advisor:	Valdis Berzins
Co-Advisor:	Luqi

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2001	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Integrated Development Environment (IDE) for Construction of a Federation Interoperability Object Model (FIOM)			**5. FUNDING NUMBERS** ARO DMSO NAVSEA
6. AUTHOR(S) Brent P. Christie and Paul E. Young			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT *(maximum 200 words)*

Advances in computer communications technology, the recognition of common areas of functionality in related systems, and an increased awareness of how enhanced information access can lead to improved capability, are driving an interest toward integration of current stand-alone systems to meet future system requirements. However, differences in hardware platforms, software architectures, operating systems, host languages, and data representation have resulted in scores of stand-alone systems that are unable to interoperate properly.

Young's Object Oriented Model for Interoperability (OOMI) defines an architecture and suite of software tools for resolving data representational differences between systems in order to achieve the desired system interoperability. The Federation Interoperability Object Model (FIOM) Integrated Development Environment (IDE) detailed in this thesis is a toolset that provides computer aid to the task of creating and managing an interoperable federation of systems.

This thesis describes the vision and requirements for this tool along with an initial prototype demonstrating how emerging technologies such as XML and Data Binding are utilized to capture the necessary information required to resolve data representational differences between systems. The material presented in this thesis has the potential to significantly reduce the cost and effort required for achieving interoperability between DoD systems.

14. SUBJECT TERMS Command, Control, and Communications, Computing and Software			**15. NUMBER OF PAGES**
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

INTEGRATED DEVELOPMENT ENVIRONMENT (IDE) FOR CONSTRUCTION OF A FEDERATION INTEROPERABILITY OBJECT MODEL (FIOM)

Brent P. Christie
Major, United States Marine Corps
B.S., State University of New York College at Buffalo, 1990

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Paul E. Young
Captain, United States Navy
M.S., University of Mississippi, 1985

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2001

Authors: [illegible]
Brent P. Christie

Paul E. Young
Paul E. Young

Approved by: Valdis Berzins
Valdis Berzins, Thesis Advisor

[illegible]
Luqi, Co-Advisor and Chair, Department of Software Engineering

[illegible]
Chris Eagle, Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Advances in computer communications technology, the recognition of common areas of functionality in related systems, and an increased awareness of how enhanced information access can lead to improved capability, are driving an interest toward integration of current stand-alone systems to meet future system requirements. However, differences in hardware platforms, software architectures, operating systems, host languages, and data representation have resulted in scores of stand-alone systems that are unable to interoperate properly.

Young's Object Oriented Model for Interoperability (OOMI) defines an architecture and suite of software tools for resolving data representational differences between systems in order to achieve the desired system interoperability. The Federation Interoperability Object Model (FIOM) Integrated Development Environment (IDE) detailed in this thesis is a toolset that provides computer aid to the task of creating and managing an interoperable federation of systems.

This thesis describes the vision and requirements for this tool along with an initial prototype demonstrating how emerging technologies such as XML and Data Binding are utilized to capture the necessary information required to resolve data representational differences between systems. The material presented in this thesis has the potential to significantly reduce the cost and effort required for achieving interoperability between DoD systems.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF SYMBOLS, ACRONYMS, AND/OR ABBREVIATIONS

CCR	Component Class Representation
COE	Common Operating Environment
DII	Defense Information Infrastructure
DISA	Defense Information Systems Agency
DoD	Department of Defense
DMSO	Defense Modeling and Simulation Office
DTD	Data Type Definition
FCR	Federation Class Representation
FE	Federation Entity
FEV	Federation Entity View
FIOM	Federation Interoperability Object Model
GMT	Greenwich Mean Time
GNP	Gross National Product
GUI	Graphical User Interface
IE	Interoperability Engineer
IDE	Integrated Development Environment
LMT	Local Mean Time
MGRS	Military Grid Reference System
OOAD	Object-Oriented Analysis and Design
OOMI	Object Oriented Model for Interoperability
RWE	Real-World Entity
SAX	Simple API for XML
SRS	Software Requirements Specification
TBD	To Be Determined
TC	Translation Class
UML	Unified Modeling Language
USMC	United States Marine Corps
USN	United States Navy
W3C	World Wide Web Consortium
XML	Extensible Mark-up Language
XPath	XML Path Language
XSL	Extensible Stylesheet Language
XSL-FO	XSL Formatting Objects
XSLT	XSL Transformations

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First, I would thank God for answering my prayers and truly blessing me here at Monterey with my beautiful new daughter Alexandria and my Masters Degree. All my knowledge and wisdom flows from him. Second, I thank my wife and best friend Maggie for making life truly worth living. And lastly, I owe a debt of gratitude to my fellow students and instructors who have made my time here at NPS a very pleasant and memorable experience.

-- Brent P. Christie

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PURPOSE

The purpose of this thesis is to develop the requirements and to build an initial prototype of an integrated development environment (IDE) used to create an interoperable federation of systems. This thesis advances the vision of Young et. al. for *Using an Object Oriented Model for Resolving Representational Differences between Heterogeneous Systems* proposed in [YBGL01]. Specifically, the objective of this thesis is to develop the requirements for a tool that will be used to construct and maintain a Federation Interoperability Object Model (FIOM) from a number of autonomously developed systems. The FIOM produced by this IDE, hereafter termed the FIOM IDE, will be used to automatically resolve representational differences between components of the federation.

In this chapter we will analyze the problem and discuss the motivation behind our effort, namely:

- Discuss the problem targeted by Young et. al.
- Ascertain the root causes of the problem.
- Identify the stakeholders and the users.
- Define the solution system boundary.
- And, recognize the constraints to be imposed on the solution.

Chapter II details Young's Object Oriented Model for Interoperability (OOMI), which provides the architectural components from which the FIOM is built. The challenges of building a flexible, extensible, and scaleable middleware system that allows efficient data integration and semantic interoperability are covered.

Chapter III contains a review of techniques and technologies proposed to implement the FIOM IDE. The advantages and disadvantages of various state-of-the-art technologies are detailed and commercially available tools are mentioned.

Chapter IV details the requirements process and focuses on the vision document and the high-level software requirements specification (SRS) for the construction of the FIOM IDE. The status of FIOM IDE prototype is also covered.

Chapter V provides concluding remarks about the current system state. Work to be done and potential for further research is mentioned.

B. PROBLEM

How do we share information between two or more system specific representations of the same real-world object? Furthermore, how do we extend the object-oriented model to handle these representational differences while maintaining all the benefits of the original model? And lastly, how do we connect the systems targeted for integration in an efficient, scaleable, and manageable manner? The FIOM is the answer to the first two questions and the FIOM IDE is the answers to the last.

The FIOM is an instance of the generic Object-Oriented Model for Interoperability (OOMI) proposed by [YBGL01]. Formally, the OOMI solves the interoperability problem by capturing the differences in representation used by autonomously developed components to describe the real-world entities (RWEs) that define the interoperation between systems. The OOMI also captures those translations needed to resolve these differences in representation. This new model allows a simple, scalable, and maintainable means to share information between system specific representations that are contained within a single RWE or across multiple related RWEs.

The FIOM IDE is a tool that will provide a computer-aided methodology that will automate the resolution of data representational differences between systems targeted for integration. The inputs to this tool are the schemas that define the targeted system's external interface. Correspondences between different system's external interfaces are used to create a new, or add to an existing, FIOM. A middleware translator then uses the information contained in the FIOM to enable interoperability between component systems. The translator will process messages based on the schemas that were used as input into the tool.

C. ROOT CAUSE

The Defense Information Infrastructure (DII) Master Plan states "Information Technology infrastructure is composed of many disparate underlying computing configurations, designed and implemented at different times to meet different requirements." Given the varying perceived service-specific requirements of DoD users and the distributed execution authority within DoD, it is reasonable to assume that heterogeneity of our systems will continue to be prevalent [DII98].

Heterogeneity, although not necessarily a negative, is something we must deal with if we are to control costs and realize the full potential of our investments. Increased costs are due to two factors. First, many of these systems unintentionally produce similar or complementary information. Second, advances in computer communications and networking technologies, a shrinking defense budget, and an increase in joint and combined operations now make it possible and necessary to integrate. To be blunt, the funding is simply not there to procure new integrated network-centric systems to replace the thousands of DoD legacy systems currently operating and produced under the stand-alone / closed-system paradigm.

The proliferation of these non-interoperable legacy systems has imposed delays, costs, and friction in nearly every area of human endeavor. By conservative estimates, about 2 percent of the gross national product (GNP) can be attributed to inefficiencies such as redundant information entry, data conversion and system incompatibilities [SB01]. The United States, having a GNP greater than 9.8 trillion dollars in 2000 [OECD01], wasted at least 197 billion dollars on these inefficiencies. To put this in perspective, these inefficiencies amount to more than double the total same year budget of the Navy and Marine Corps combined [DL99].

The cost of not integrating is overwhelmingly evident. As the defense budget shrinks, these inefficiencies will take a larger slice of the total and result in less funding needed to strengthen our warfighting capabilities. A history lesson on the reasons why this came about is not appropriate for this paper nor would add any insight on a solution. What is important is that the DoD must move to meet the computing goal of evolvability and interoperability within this heterogeneous environment in the most intelligent and cost efficient manner.

The current method for integration of legacy systems involves a careful, time-intensive study of a producer system's exported information. Using this study, the system integrator then determines if this information is of value to a consumer system and what formatting or transformations are required. After mapping the exported data type to the imported data type, the system integrator then writes a software "wrapper" which will perform a translation from the producer's representation to the consumer's. Unfortunately, this process must be repeated for every new piece of information a consumer or producer would like to make available to the system. Currently there is little or no computer assistance for performing this task. [Lytt00]

The approach proposed by Young et. al. [YBGL01] is a more efficient means of accomplishing systems interoperability and will allow the DoD to lower costs three ways. First, by isolating the internal code of these systems with a "wrapper." Viewing each component system as a black box and only dealing with the interfaces allows connections without understanding or modifying the internal components of a system. This interface is the input to our tool and the development of it is outside the scope of this project.

Second, each component system only has a single interface (intermediate representation) to the FIOM vice multiple interfaces (point-to-point) as with the current method described above. For example, with two systems wanting to share one piece of data each it takes at most 2 translations. But, when connecting N systems it takes at most $N^2 – N$ translations. With the FIOM, that same scenario would only take at most 2N translations. Exponential savings is realized as the number of systems connected grows. Also, if a component system changes, only one interface to the FIOM needs to be updated vice up to N-1 interfaces with other component systems. This can be seen in Figure I-1 below. This intermediate representation technique that the FIOM represents will have an exponential savings effect when compared with the point-to-point method.

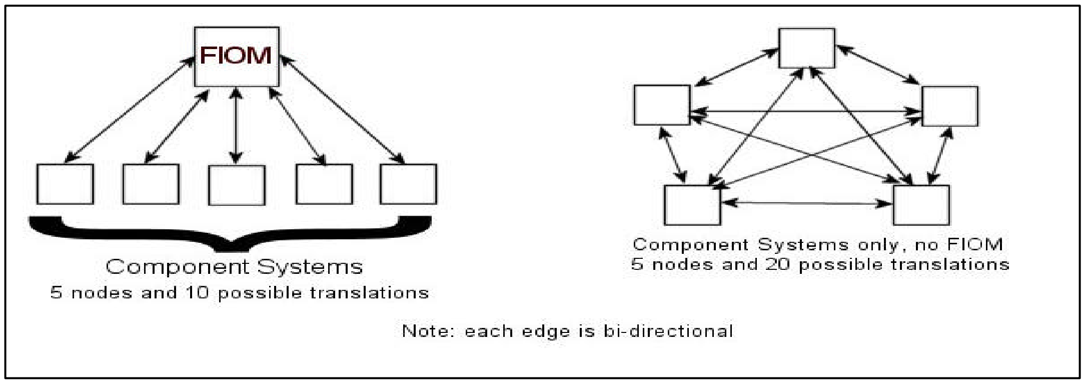

FIOM
Component Systems
5 nodes and 10 possible translations
Component Systems only, no FIOM
5 nodes and 20 possible translations
Note: each edge is bi-directional

Figure I-1. Translations required with and without FIOM.

The benefits of this move toward interoperability and using a tool like the FIOM IDE include:

- Reduction in operating costs, thus freeing monies and giving more flexibility in making funding choices for other programs.
- More timely and efficient integration of an arbitrary legacy system into an existing or newly created federation.
- Ability to easily add new systems to a federation without impacting existing systems in the federation.
- A revolutionary and synergistic effect on how our systems and warfighters communicate and share information.
- And ultimately, a leaner and stronger military to fight and win our next war.

D. USERS AND STAKEHOLDERS

The main user of this tool is an interoperability engineer (IE). An IE is defined as an individual that is an experienced Software Engineer that is responsible for integrating DoD systems. It is assumed that the IE is not necessarily a domain expert in the particular systems that are to be integrated. But, the IE is expert in the federation namespace being used to create the FIOM. Once the FIOM is created it will be used within a network to convert messages between component systems.

The main stakeholder of the system is the United States Department of Defense. It is our intention to provide a proof of concept to allow evaluation, further research, and possible implementation of our work.

E. SOLUTION SYSTEM BOUNDARY

A black box view of component systems is taken in creating a federation of interoperable systems. All data that enters or exits a system via messages is eligible for integration once it is formatted as an Extensible Markup Language (XML) document with a corresponding XML Schema. Further discussion on what XML is and the choice to have data in this format is discussed in chapter 3. A component system XML Schema is taken as input into the FIOM IDE. It is then processed and matched to an existing or newly created Federated Entity (FE) within the FIOM. FEs are descriptions of RWEs within a particular FIOM. Figure 1-2 provides a simplified system perspective for the

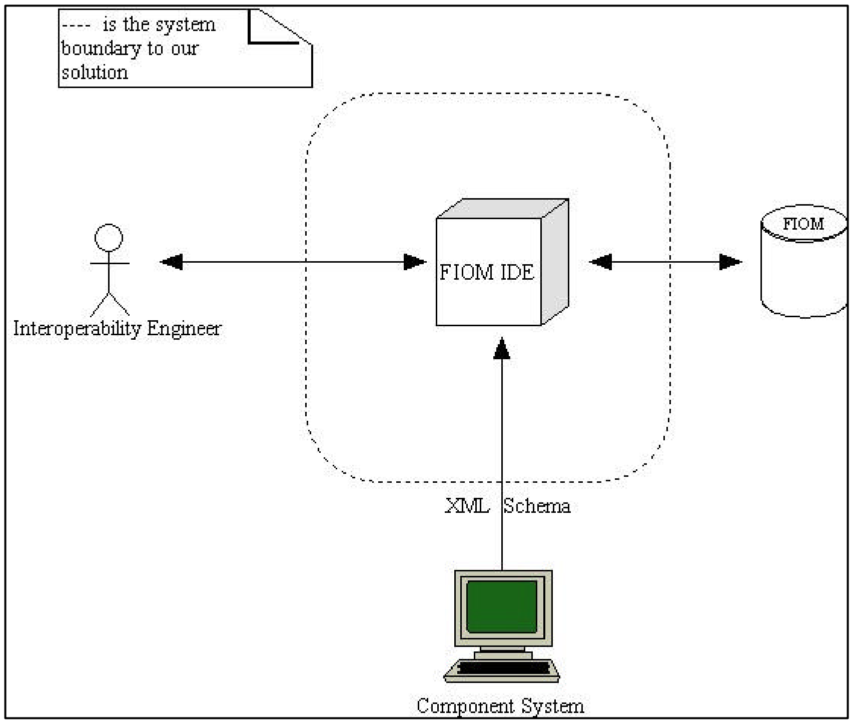

---- is the system boundary to our solution
Interoperability Engineer
FIOM IDE
FIOM
XML Schema
Component System

Figure I-2. FIOM IDE perspective

FIOM IDE. The FIOM is the main output of the FIOM IDE and will be used by a middleware translator application to allow interoperability of component systems as shown in Figure 1-3.

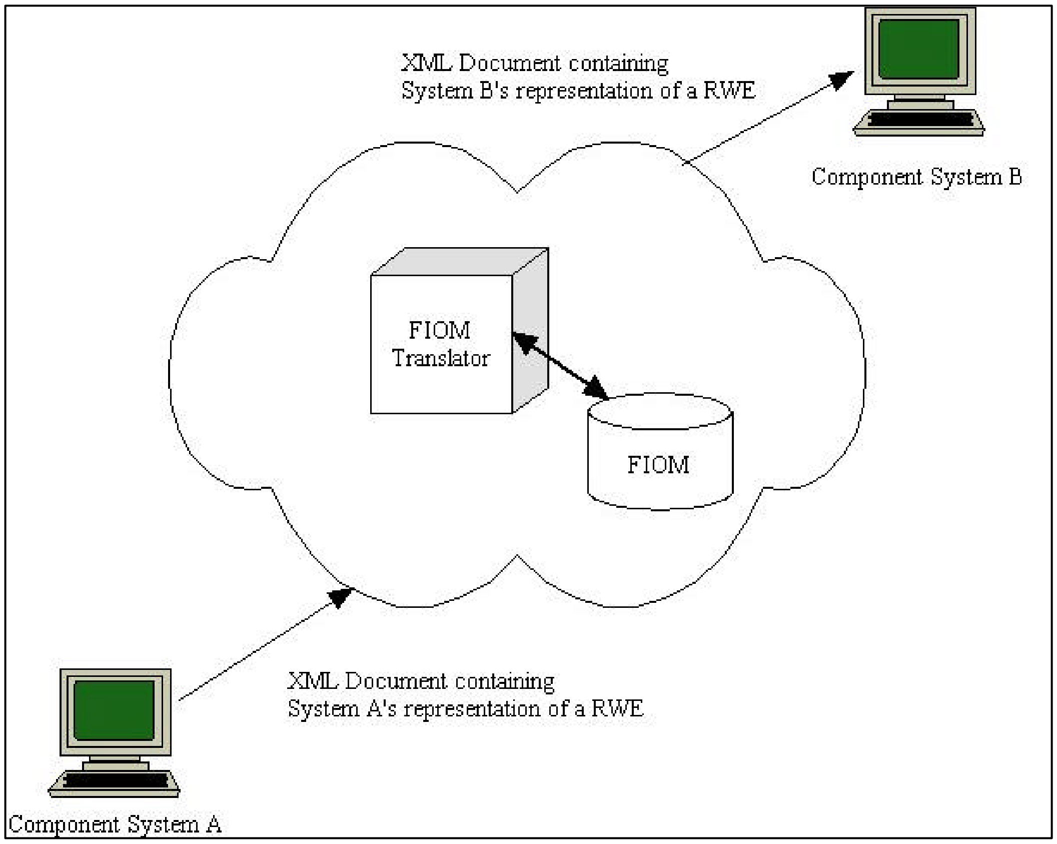


Figure I-3. Translator perspective.

The translator can be located anywhere between systems that wish to share data. The translator can be a wrapper around Component System A, around Component System B, somewhere in between, or a variation of all three. Design of the FIOM IDE must take into consideration the FIOM Translator to ensure no conflicting requirements are developed.

F. CONSTRAINTS TO BE IMPOSED ON THE SYSTEM

All of our constraints are self-imposed to limit the scope of this project and are as follows:

- Extensible Markup Language (XML) will be used for data integration. Again, the merits of XML will be discussed in chapter 3.
- All component systems will be viewed as black boxes as seen in Figure I-4 below. All data input and output targeted for integration will have an XML Schema to be used as input into our tool. Once a component system is integrated into the FIOM, the component system will pass messages in the form of an XML document based on the corresponding input schema.

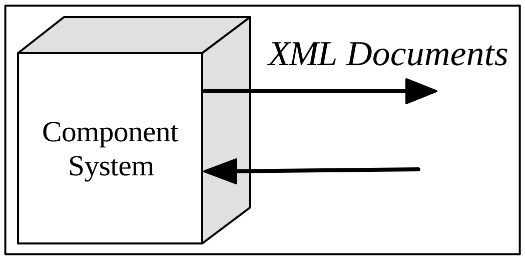


Figure I-4. Black Box View of Component System

- It is assumed that the XML Schemas for component systems will not have to be developed by the IE. Consideration on how these schemas should be developed can be found in [Young02].
- Java will be used to develop the prototype of the FIOM for two reasons. First, Java is the best language for integration with XML. Second, most of the open-source XML browsers, editors, parsers, validators, and data-binding tools are written in Java. This wealth of open-source material will expedite the building of the FIOM IDE prototype.

II. OBJECTS, INTEROPERABILITY, AND MODELS, OOMI!

A. INTRODUCTION

This chapter provides a summary of the Object Oriented Model for Interoperability (OOMI) introduced by Young et. al. in [YBGL01]. Knowing the OOMI, and the theory behind it, will help in understanding the design issues of creating the tool used to support the construction and maintenance of an OOMI instance, the Federation Interoperability Object Model (FIOM). This knowledge will aid the reader in analyzing the issues discussed in later chapters.

B. OBJECTS

In contemporary object-oriented modeling, an object is a software representation of some real-world entity in the problem domain. An object has identity (i.e., it can be distinguished from other objects by a unique identifier of some kind), state (data associated with it), and behavior (things you can do to the object or that it can do to other objects). In the Java Programming Language these characteristics are captured as the name, member variables, and methods of a class, respectively. [YBGL01]

This view of objects and classes has proven valuable in the development of countless systems in various problem domains encompassing all degrees of size and complexity. However, one common characteristic of the majority of these object-oriented developments is that they were produced by a development team that shared common objectives and had a common view of the real-world entities being modeled. Most projects also involve a common architecture implemented on a common target platform, using the same implementation language and operating system. As a result, a single scheme for depicting an entity's name, attributes, and operations as well as the means for representing these properties has been the norm. Therefore, capturing the representation of these properties has not been an issue. The software representation of the real-world entity should have the same name, attributes, and operations across all elements of the architecture if the development team enforces consistency. [YBGL01]

This is not necessarily the case when integrating independently developed systems. The different perspectives of the real-world entity being modeled by independent development teams will most likely result in the use of different class names as well as differences in the number, definition, and representation of attributes and operations for that same real-world entity. These representation differences must be reconciled if the systems are to interoperate. [YBGL01]

Young has developed an object-oriented model for defining the information and operations shared between systems. The initial use of the model is targeted for integration of legacy systems, which generally have not been developed using the object-oriented paradigm. However, defining the interoperation between systems in terms of an object model provides benefits in terms of the visibility and understandability of the shared information and provides a foundation for easy extension as new systems are added to an existing federation. The object model defined in this chapter can be easily constructed from the external interfaces defined for most legacy systems (whether object-oriented or not). [YBGL01]

Section C categorizes the representational differences that exist in autonomously developed systems. Section D & E introduces the Object-Oriented Model for Interoperability (OOMI) as a means for capturing the information required for resolving these representational differences. Section F introduces an automated environment for constructing an instance of the interoperability object model for a federation of systems, the FIOM Integrated Development Environment (FIOM IDE). Section G presents an overview of the use of the resultant by a wrapper-based translator for enabling interoperability among legacy systems.

C. INTEROPERABILITY

Variations in the representation of a real-world entity on different systems can be classified as falling into one of two general categories. The first difference is in the information utilized by each component system to represent the entity. Termed *heterogeneity of scope*, this refers to the fact that differing amounts and types of information can be captured by various systems to represent the state and behavior of an entity [Wie93].

For example, suppose a federation of four autonomously developed military systems contained information about an enemy surface-to-surface missile launcher. Because independent development teams created them, each system provides a different perspective on what state and behavior information should be contained in a model of that real-world entity. As can be seen from Figure II-1, each system includes different aspects of the entity's state. For instance, systems A and D include information about the missile system's type, position, and time. System B captures position, time and range information on the entity, and System C utilizes type, position, time, and range to describe the missile system. Similarly, each system could capture different aspects of the behavior of an entity. These differences in the state and behavior used by a component system to characterize a real-world entity can be thought of as providing different *views* of the entity by the systems concerned. [YBGL01]

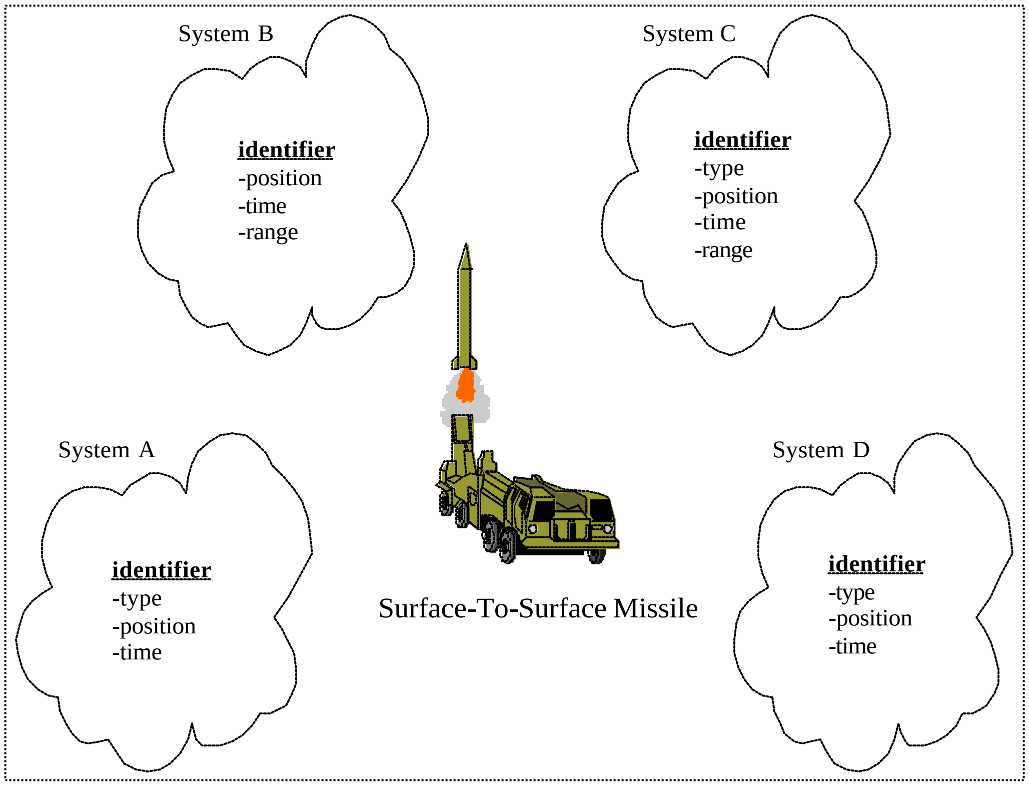


Figure II-1. Differing Views of Real-World Entity. From Ref. [YBGL01]

Even if two systems provide the same view of the entity being modeled, that is they both contain the same state and behavior information about the entity, there may still be differences in the representation of that information on different systems. This *heterogeneity of representation* [Wie93] refers to differences in the terminology used, format, accuracy, range of values allowed, and structural representation of the included state and behavioral information [KM98]. This difference in representation is illustrated in Figure II-2 by systems A and D. Even though these systems both have the same view of our real-world entity, i.e., both capture the type, position and time for the entity; they each represent the information comprising that view in a different manner. For example, System A refers to our entity as a *MissileSystem* and names its type attribute *missileDesignation*. System D refers to our entity as a *MissileLauncher* and names its type attribute *missileType*. Additionally, System A captures the entity *position* in latitude/longitude coordinates and *time* using Greenwich Mean Time (GMT) as the reference, whereas System D records entity *location* using Military Grid Reference System (MGRS) coordinates and records *time* using Local Mean Time (LMT). Figure II-2 illustrates the different views of our example real-world entity and the various representations provided for each view. [YBGL01]

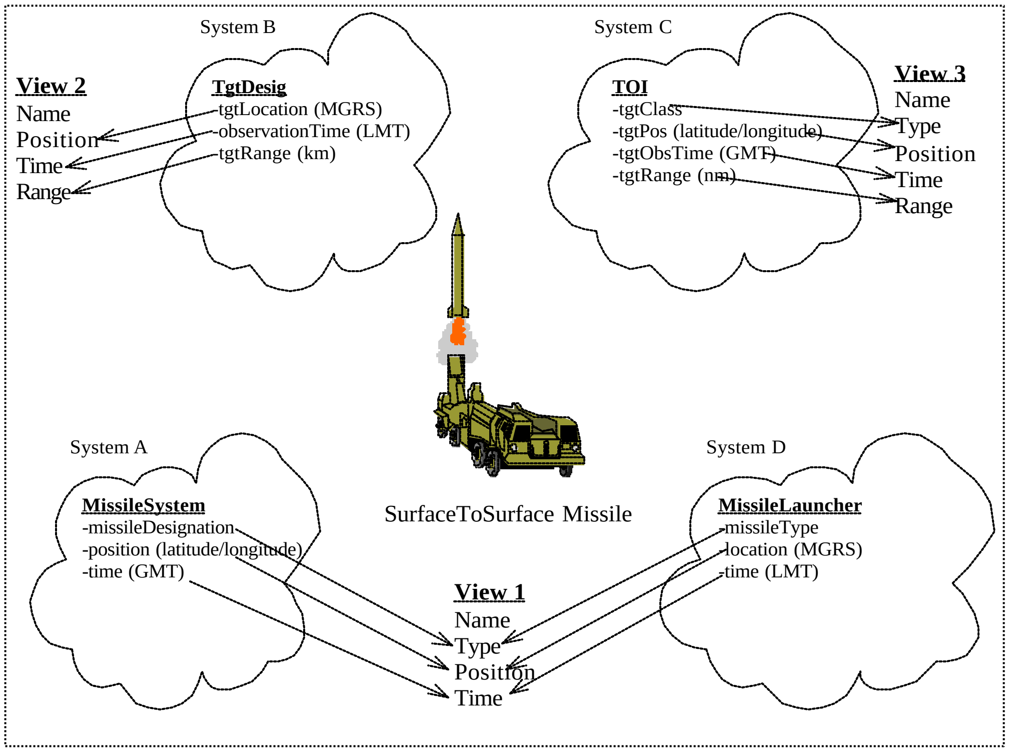


Figure II-2. Differing Real-World Entity View Representations. From Ref. [YBGL01]

The goal of Young's research is to provide a computer-aided methodology to aid in the resolution of differences in the representation of data between systems targeted for integration in order to enable system interoperability. Pitoura defines interoperability as the capability of systems to exchange information and to jointly execute tasks [Pit97]. The information exchanged between interoperating systems consists of data associated with the real-world entities being modeled by systems of the federation. The joint execution of tasks reflects the capability of an entity on one system to employ the services of an entity on another. Thus, interoperation can be characterized in terms of the real-world entities whose state and behavior are shared between systems in a federation. This thesis will only focus on Young's work in the exchange of information. Young's research on the joint execution of tasks is still ongoing and therefore will not be discussed. As stated previously, there can be differences in view and representation of these real-world entities. In order to achieve interoperability, a means for resolving these differences in view and representation is needed. [YBGL01]

D. MODEL

Young provides a means for resolving these differences in view and representation with his simple, yet powerful, model. Principal objectives of the model are:

- To clearly depict the real-world entities whose information are shared between systems in a federation.
- To provide computer aid to the process of determining the differences in view and representation of those entities.
- And, to provide automation support for defining the translations necessary to resolve representational differences between systems.

In order to achieve the objectives outlined above, the model should:

- Provide an abstract representation of the real-world entities whose information is shared between component systems, hiding the details of how that information is represented on different systems.
- Capture the different views component systems might have of the real-world entities that represent shared information. In addition, the model should capture differences in representation of those views among component systems.
- Contain information needed to aid the Interoperability Engineer (IE) by providing computer assistance to the discovery of the real-world entities that define the information being shared between systems in the federation.
- Enable identification of allowable information sharing between component systems, and contain translations, where required, to resolve differences in representation of information shared between systems.
- And, be extensible; adding new component representations of real-world entities that define shared information or including new information to be shared between systems should not affect contents or relationships in the existing model. [YBGL01]

In evaluating the objectives and goals outlined above, Young determined that an object-oriented approach offered the greatest promise for satisfying these requirements. Object-Oriented Analysis and Design (OOAD) provides principles of abstraction, information hiding, and inheritance that can be employed to meet the specified goals and objectives [KA95, WCS00]. However, conventional use of these object-oriented principles and techniques is not sufficient for resolving representational differences between heterogeneous components of a system federation. Instead, a model-based approach built on OOAD principles is presented to satisfy the requirements for heterogeneous system interoperability [YBGL01]. The resulting model, the *Object Oriented Model for Interoperability (OOMI)*, is described next.

E. OOMI

1. Capturing Real-World Entities and Views

It is expected that for a federation of heterogeneous systems, a number of real-world entities (RWEs) will be involved in the interoperation between systems. Under the OOMI, the collection of RWEs used to define the interoperation of a specified federation of systems is termed a *Federation Interoperability Object Model (FIOM)*. All of the normal relationships between classes, packages, interfaces, and other elements used in the OOAD paradigm are available for use with federation entities in the FIOM.

Real-world entities within the FIOM are captured using the concept of a *Federation Entity (FE)*. The FE provides a level of abstraction representing the information being shared between ***different but related*** component systems while hiding the details of how that information is being represented on different systems. For each FE, another level of abstraction, called a *Federation Entity View (FEV)*, is used to distinguish the variations in the information used for representing the ***same*** real-world entity on different systems [YBGL01]. Figure II-3 illustrates the OOMI archetype for a real-world entity (FE) which shows how it may contain several different views (FEVs).

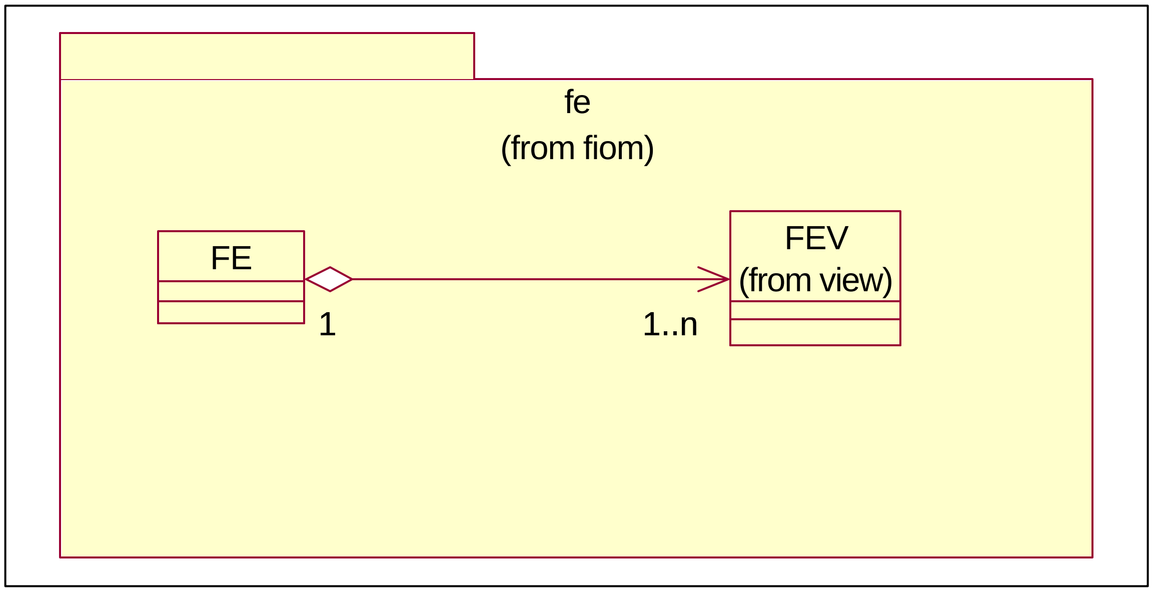


Figure II-3. OOMI Real-World Entity Archetype

2. Capturing Federation Entity View Representations

As discussed earlier, different component system implementations of the same RWE may result in variations in the terminology, definition, and representation of the attributes defined for that RWE. In order to resolve these differences, the OOMI provides two mechanisms to capture the possible alternative representations of an entity's view. The first mechanism, a *Component Class Representation (CCR)* is a special-purpose class used to capture the alternative ways various component systems may represent a view of a FE, i.e., a FEV. A CCR will be defined for a view only when a class defined in the external interface of a component system exhibits a one-to-one correspondence between the attributes of the component class and those properties contained in the view. As will be shown later, there may be views defined for a FE that do not have a corresponding CCR as well as views containing multiple CCRs.

It is reasonable to assume that most leagacy systems were not developed using object-oriented design methodogy. Therefore, component system representations may not necessarily be in a useable object-orientative form. If this is the case, a *transformation* is required for proper correspondence. A *transformation* is taking a component system representation of a RWE and creating a CCR, which contains all the information, in object-orientative terms, of that RWE. The component system's domain expert should handle most, if not all, of the transformation process. In the case of our current prototype of the FIOM IDE, the transformation is a two-part process. First, XML Schemas, representing component system RWEs, are created and received from a domain expert. Next, the schemas are *transformed* into CCR classes via data-binding. At runtime, XML Documents, representing component system RWEs instances, are *transformed* into CCR instances via unmarshal and marshal methods. These methods are added to the CCR class as seen if Figure II-4. Unmarshal, marshal, and data-binding will be discussed in detail in Chapter III.

In order to take advantage of the reduced number of translators required with the use of the intermediate representation approach (see, chapter 1), the OOMI adds a second special-purpose class to a FEV, the *Federation Class Representation (FCR)*. The FCR is used to encapsulate the "standard" representation used by the federation for a particular view of a real-world entity. Each FEV will contain exactly one FCR representing this "standard" representation of the view [YBGL01]. Figure II-4 provides a depiction of a FEV with component FCR and CCR.

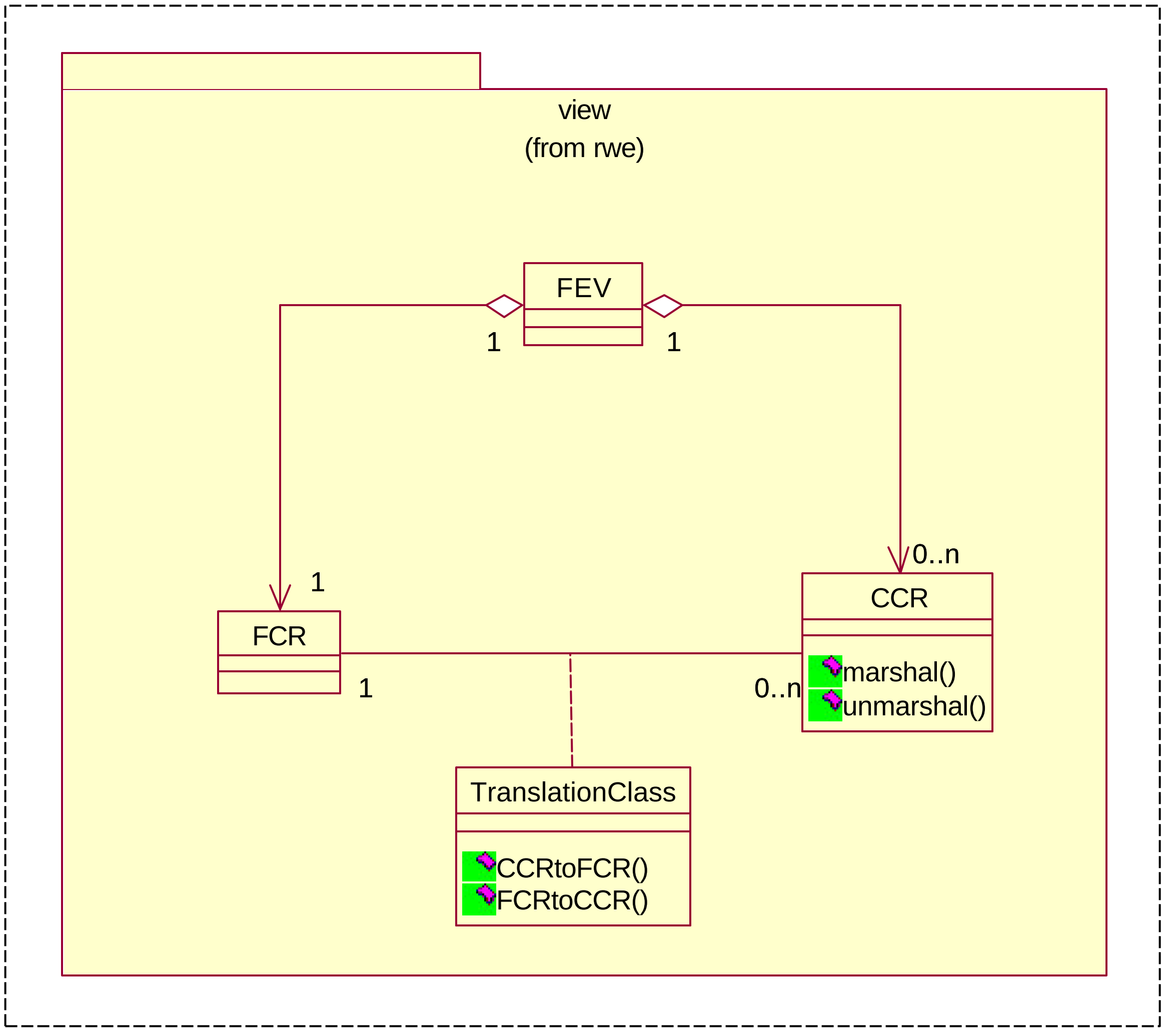


Figure II-4. Federation Entity View

3. Sharing Information between Component Systems of the FIOM

a. Resolving Differences in View Between Systems

Rarely will two different systems' view of a federation entity be identical. In order to share information between two systems that have different views of the entity(s) defining the interoperation, these differences in view must be resolved. Fortunately it is just as rare that different systems' views of an entity are mutually exclusive (otherwise they wouldn't be able to interoperate). Generally, two or more systems' view of the same entity will have some areas of commonality. Two systems' representations may capture the same core state of an entity with each including additional characteristics as required by the specific application. In this situation a view could be defined for the core state information, and separate views defined for the extended information. The views containing the extended information can be considered to be subtypes of the view containing the common core information. Commonalities in view between component system entity implementations enable us to determine when a supertype-subtype relationship exists between component system views. Then, through exploitation of the Liskov and Wing notion of behavioral subtyping [LW94], we can determine when the information contained in one system's view of an entity is suitable for use by another. As recapitulated by Wing and Ockerbloom, "If S is a subtype of T, users of T objects cannot perceive when objects of type S are substituted for T objects." [WO00 p.579] When applied to the context of the OOMI, a component system can always utilize objects that are an instance of the subtype of the federation entity view defined for that component [Young02].

A FCR can share information via the FEV abstraction due to the extension inheritance structure it is a part of. As seen in Figure II-5, this extension inheritance structure allows a FCR to share information between itself and any ancestor it has. Information can be shared "up" the inheritance hierarchy through promotion without loss. In contrast, an ancestor's information shared "down" through demotion will be inadequate. Thus a system whose perspective of a Tank is reflected by FEV 0 in Figure II-5 could utilize information from systems whose view of a tank was captured by any of Tank FEVs 1 through 4. Young is currently working on way to remedy this "up only" sharing. The research on the remedy is not complete but it involves determining if the information that extends an ancestor within a child has default values or not.

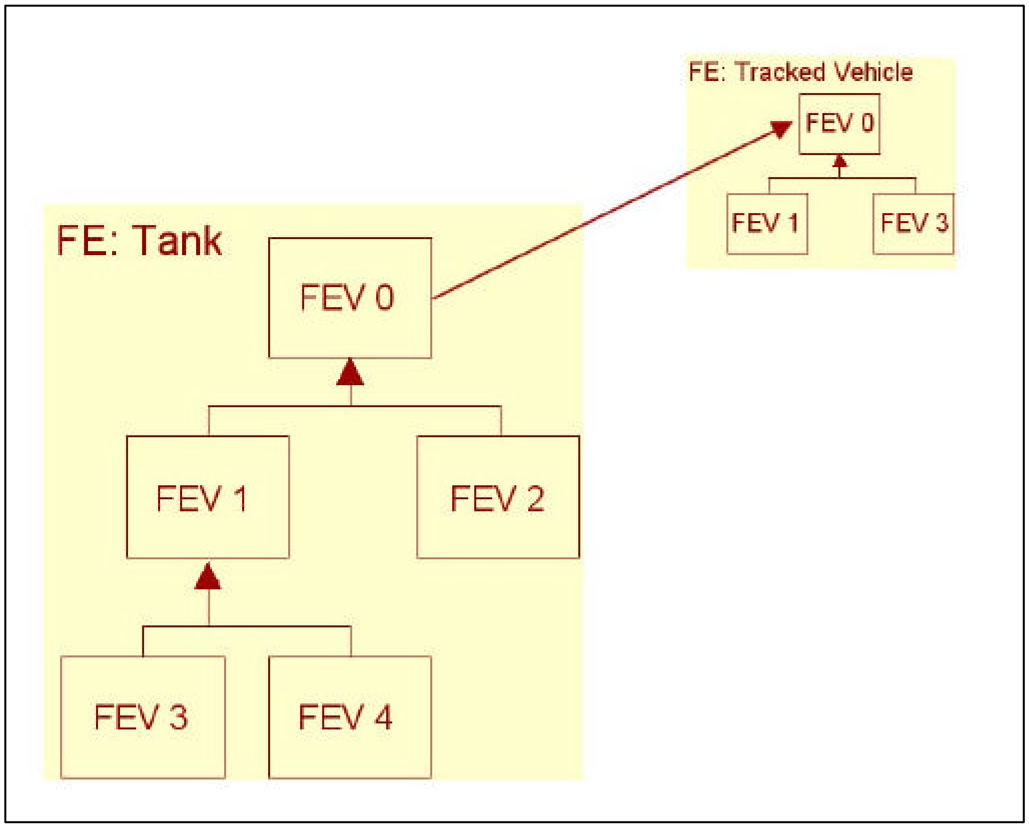


Figure II-5. Inheritance Relationships Within The FIOM

Figure II-5 also shows how the FE allows sharing information between related RWEs. As depicted by the figure, a Tank is a subtype of a Tracked Vehicle. Therefore information about a Tank would be substitutable in a system expecting information about a Tracked Vehicle. The relationship between FEs in the FIOM is restricted to the root FEVs within each FE. The root FEV, and namely the FCR within it, represents the minimum amount of information that describes an entity. Allowing only inheritance relationships between root FEVs removes the possibility of multiple inheritance and simplifies the model. This simplicity may equate to loss of information but the gains in managing and understanding the relationships within the FIOM far outweigh this loss.

b. Resoving Differences in Representation

The power of the relationship between a FCR and a CCR is that, when translating representations between a FCR and a CCR from the same FEV, there is no loss of information. This means the relationship between FCR and CCR is reflexive and preserves "round tripping." Although a FCR and CCR from the same FEV contain the same information, they may represent that information differently, i.e. each may use different terminology, data types, data accuracies, etc. These differences in representation must be resolved. This is accomplished with an association class called a Translation Class (TC), as seen Figure II-4. As the name implies, the TC is responsible for taking the information encapsulated in a CCR and handling the functional translations necessary to map the information, without loss, into a FCR, and vice versa. An example of this translation can be seen in Figure II-2. The figure shows how System A and D are mapped into View 1. Notice the difference in naming conventions and type information of member variables contained in System A's and D's representation when compared to View 1. Dealing with these differences is the most difficult aspect of interoperability. This was the impetus in creating the FIOM IDE, which automates as much of this process as possible. Further reading on the translation process can be found in [Young02].

To further simplify the FIOM model, the FCR representation is based on an *ontology* containing the federation-sanctioned description of an entity's meaning. This ontology can be developed specifically for a federation of systems or it can be derived from a domain-specific or industry-wide standard. Two possible standards to draw from are the Defense Information Systems Agency's (DISA's) Defense Information Infrastructure Common Operating Environment (DII COE) XML Registry and the Defense Modeling and Simulation Office's (DMSO's) Functional Description of the Mission Space (FDMS) namespaces [DII01, FDM01]. While multiple namespaces may be present within the FIOM, naming conventions, based on an ontology, must and will be enforced within the FIOM. This will allow the sharing of information between different FEVs within a FE, and between FEs themselves, using simple assignment operations.

F. CONSTRUCTING THE FIOM

Enabling a collection of related software systems to share information has the potential for significantly enhancing the capability of the resultant federation of systems over that of the individual components. The previously introduced OOMI is used to enable information sharing among a federation of autonomously developed heterogeneous systems. Using the information contained in the OOMI, computer aid can be applied to the resolution of data representational differences between heterogeneous systems. In order to apply computer aid, a model of the real-world entities involved in the interoperation, termed a Federation Interoperability Object Model (FIOM), is constructed for the specified system federation. Construction of the FIOM is done prior to run-time by an Interoperability Engineer (IE) with the assistance of a specialized toolset, called the FIOM Integrated Development Environment (FIOM IDE).

The Graphical User Interface (GUI) based FIOM IDE is used to:

- Discover the information and operations shared between federation components.
- Provide assistance in identifying the different representations used for such information and operations by component systems.
- Define the transformations required to translate between different representations.
- And, generate system-specific information used to resolve representational differences between component systems.

The first task in FIOM construction is determining the real-world entities that are shared between systems in the federation. In other words, one system has a real-world entity to offer and another system wants it. Determination of the real-world entities that define the interoperation of a federation is not merely a matter of identifying the classes contained in the external interfaces of the component systems. Because of the independently developed, heterogeneous nature of the systems in the federation, each system may have a different representation for the real-world entities involved. Identifying which of a component system's classes are representations of the same real-world entity is a key step in achieving interoperability between the component systems. Correlation software is included as part of the FIOM IDE in order to assist the IE in this effort. The correlation software searches the FIOM for a matching FCR to the component system's CCR. If none within the returned set of possible matches are acceptable, the IE is given an opportunity to create a new FCR by adding a new FEV and/or a new FE. The information that makes correlation possible is captured in the syntax and semantic class of the FCR and CCR as seen in Figure II-5. A detailed discussion on the correlation process can be found in [Pug01] and [Young02].

After identifying or creating a matching FCR for a CCR, the transformations required to translate between the different representations must be defined. The FIOM IDE assists the interoperability engineer in this task through the use of a GUI-based matching process used to provide computer aid to translation development, and the maintenance of a translation library to enable the reuse of common translation algorithms.

Finally, class transformation and relationship information is extracted from the FIOM for each component system. The system-specific information is used by a wrapper-based translator to resolve representational differences between component systems.

G. USING FIOM TO RESOLVE REPRESENTATIONAL DIFFERENCES BETWEEN HETEROGENEOUS SYSTEMS

When information exchange takes place between heterogeneous systems, the interoperability object model constructed during the *pre-runtime* phase for a specified federation of component systems is used to derive a translator. Differences in view and representation of information shared between interoperating systems are reconciled at *runtime* by the *translator*, which serves as an intermediary between component systems. The translation function is implemented as part of a *software wrapper* enveloping a producer or consumer system (or both) in a message-based architecture, or alternatively as part of the data store (actual or virtual) in a publish/subscribe architecture. A software wrapper is a piece of software used to alter the view provided by a component's external interface without modifying the underlying component code [YBGL01]. Figure II-5 shows an overview of the use of software wrappers and the involvement of the Federation Interoperability Object Model in the translation process.

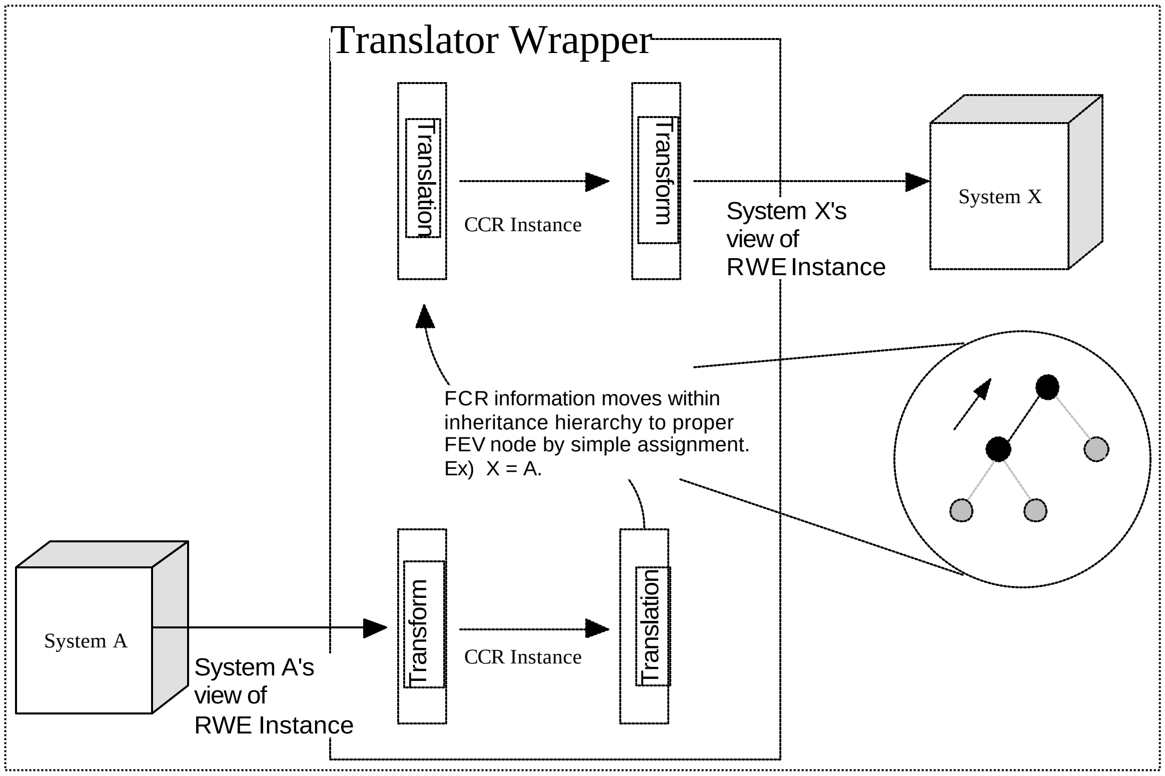


Figure II-6. Translator Wrapper

Using Figure II-6 above, an example of how this process works is as follows:

- Component system A sends a RWE containing information needed by System X.
- System A's RWE must first be transformed into an object (CCR instance). The transform mechanism is defined pre-runtime and is necessary to ensure the RWE is described in object-oriented terms. This CCR will be called CCR A.
- The CCR A is then translated into its corresponding FCR, called FCR A.
- Since System X's CCR has a corresponding FCR, called FCR X, which is an ancestor of FCR A, a simple assignment is made (FCR X = FCR A).
- FCR X is then translated to its corresponding CCR, called CCR X.

- CCR X is transformed into System X's representation of the RWE that originated from System A.

As indicated above, the translator must be capable of transforming a component system's RWE into a CCR object and then translating it to its federation counterpart (FCR), and visa versa. The information required to effect these transformations and translations is captured as part of the FIOM during federation design. Then, at run-time, the translator accesses the information contained in the model to resolve differences in federation entity view and to effect the translation between component and standard representations of a view.

THIS PAGE INTENTIONALLY LEFT BLANK

III. TECHNOLOGIC INTEROPERABILITY ENABLERS

A. INTRODUCTION

In this chapter we discuss the technologies that were selected to implement the FIOM IDE. Each discussion will focus on a description of the technology, a comparison of competing implementations of that technology, and why that particular technology was chosen. But before we continue, there are 3 major factors that drove our choices in the design of the FIOM IDE: constraints from our major stakeholder, the OOMI's technology needs, and reducing costs.

1. Directives

The DoD is counting on XML and XML-related technologies to enable information dissemination management capabilities and to solve many interoperability issues. DoD Directive 8320.1, DoD Data Administration [DoD91], authorizes the establishment of and assigns responsibilities for DoD data administration to plan, manage, and regulate data within the Department of Defense. The Defense Information Systems Agency (DISA) is designated as the lead agency responsible for executing the policy and procedures and making DoD Data Standards available to the community. DISA is using XML as its common exchange data format in support of its Defense Information Infrastructure Common Operating Environment (DII COE) data engineering strategy. The bottom line is that XML will be used to pass information between DoD systems.

2. OOMI's Technology Needs

The OOMI view of real-world entity representation demanded technologies that fully supported an object-orientated approach. We quickly realized that XML and some of its related technologies could not be used alone. Java with its strong typing, properties, methods, interface inheritance, behavior inheritance, polymorphism, reflection, and its compatibility with XML, made its use essential. Also, some XML technologies like DTD and XSLT were found to be incompatible, too immature, or too cumbersome. All choices had to pass the test, "Will this technology help us effectively implement the FIOM and the FIOM IDE?"

3. Cost

Reducing the cost of doing business is one of the major benefits of this project. In this vein we sought ways to reduce the cost of actually building the tool itself. The biggest reduction is due to the liberal reuse of open-source software. Open-source software reduces costs four ways. The first and most appealing aspect is that it's free! Second, the world can read, modify, and even redistribute this type of software. This allows us to leverage and build on the work of others. This reduces the cost of labor in producing our own code. Third, because of the second point we are assured that the software we incorporate will continue to evolve while at the same time exposing the code to testing by the masses. Again, this is a savings in labor with regard to testing and maintenance. And lastly, the life cycle for this type of software is extremely compressed, i.e., better software faster, when compared with proprietary software. It's no wonder, with so many companies and individuals all being able to contribute.

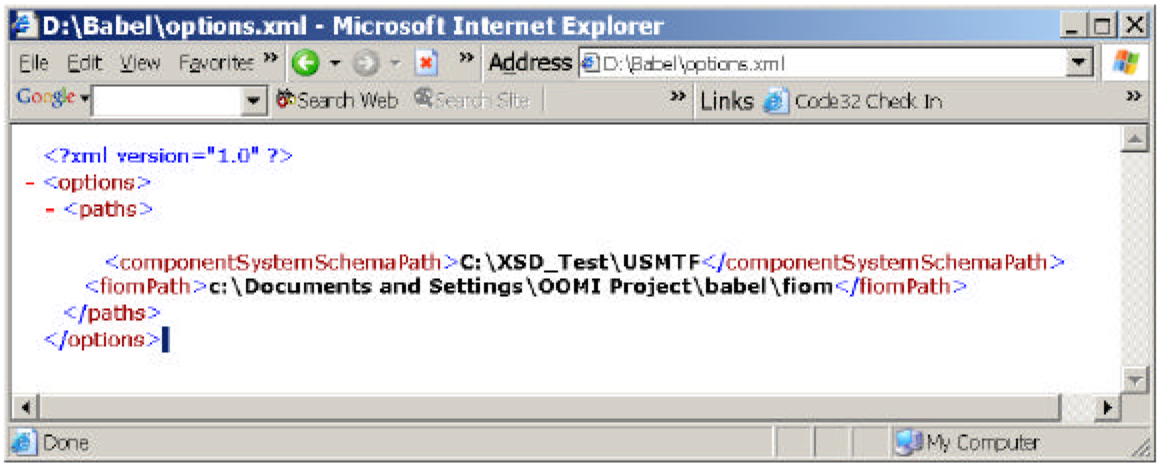
B. XML THE NEW LINGUA-FRANCA

XML stands for Extensible Markup Language, which is a meta-markup language that became a World Wide Web Consortium (W3C) recommendation in January of 1999. It is the new lingua-franca for structured documents and data on the Internet. It is platform-independent, non-proprietary, customizable, self-describing and human readable. It is not our intent to provide an in-depth discussion on XML. Further information can be found at [W3C].

XML is a family of technologies. We will focus on a few members of this family, namely: DTD, XML Schema, SAX, DOM, and XSLT. But before we begin, we would like to give the reader a quick look at what XML is and discuss the key concepts of well-formedness and validity.

1. XML basics

The best way to get a feel for XML is by viewing an example. Figure III-1 below shows a simple XML document that is used to handle the persistent storage of options for the FIOM IDE, displayed in Microsoft's Internet Explorer Version 6.0.



```
  <?xml version="1.0" ?>
- <options>
  - <paths>
      <componentSystemSchemaPath>C:\XSD_Test\USMTF</componentSystemSchemaPath>
      <fiomPath>c:\Documents and Settings\OOMI Project\babel\fiom</fiomPath>
    </paths>
  </options>
```

Figure III-1. Simple XML Document

As you can see, XML looks a lot like HTML. The major difference is that XML is a meta-language that offers no inherent clue as to how the information should look. This frees XML from the static tag set of HTML and also separates the model created from its view. All XML documents are properly nested (hierarchical) tree structures. This example document contains a root element <options>, which contains one child element <paths>, which contains two child elements called <componentSystemSchemaPath> and <fiomPath>. Notice that the XML document nicely describes the structure of the data but does not say much as to what the elements mean, other than what can be inferred by the element names.

2. Well-formedness

A document is not an XML document unless it is well-formed, i.e., syntactically correct, according to the W3C's XML specification. This means that an ill-formed document will not be accepted for processing. This simplifies the internal code of parsers and also speeds up the processing of documents.

3. Validity

An XML document is valid if it has an associated schema, DTD or XML Schema, and if the document complies with that schema. A schema further constrains the syntax of the XML document and also adds semantics through documentation within the schema itself.

C. CONSTRAINING CONTENT

1. DTD

A Document Type Definition (DTD) specifies the logical structure of the document; it is a formal grammar describing document syntax and semantics. We will not discuss DTD beyond this section because of this technology's many flaws. The DTD's most crippling flaw is that it has no capability for strong typing. DTDs treat almost all of its data as strings. Existing free text searches can't differentiate between a Marine, Marine Air Ground Task Force, Marine Corps Band, or a Marine Weather Forecast. Strong typing is need if we are going to effectively match different representations of the same RWE.

DTDs are not even XML. This disallows the use of many XML tools for displaying and manipulating information within the DTD. This would make it difficult to find open-source XML based APIs that support DTDs for use in our tool. These fatal flaws make DTDs unsuitable for our purposes.

2. XML Schema

XML Schema is much more suited for our purposes. XML Schema can specify actual data types for each element's content within a document. This is essential if we are going to have enhanced, efficient, and accurate searching capabilities within our tool. Also, XML Schemas have built in elements to handle annotations to add semantics to the types created. XML Schema can handle multiple namespaces allowing a means to process homonyms, which is also essential in matching RWEs. A more detail discussion of XML Schema can be found at [W3Csch].

D. PROGRAMMATIC ACCESS

1. SAX

Simple API for XML (SAX) and DOM (Document Object Model), which we will discuss in the next section, were both created to serve the same purpose. Their purpose is to provide access and modification capability to the information stored in XML documents using any programming language. However, both of these techniques take very different approaches in providing that access.

SAX provides access to documents as a sequence of events. It works as follows:

- A SAX parser sequentially processes an XML document, signaling an event when a specified item such as an open tag or close tag is found.
- The programmer is responsible for interpreting these events by writing an XML document handler class. This handler class is responsible for specifying what action is required to be taken when a tag is encountered, such as storing an element for future reference.

2. DOM

XML only supports "has a" or "parent-child" relationships, such as a <person> may contain sub-elements of <name>, <social_security_number>, <height>, <weight>, <eye_color>, etc. This hierarchical tree structure is preserved with the Document Object Model (DOM). DOM creates a tree of nodes (based on the structure and information in an XML document) and provides access to this information by interacting with this tree of nodes. The DOM takes a generic approach, in that it will take any arbitrary XML document and model it. Once a document object tree has been created (by the XML parser, or your own code), you can access elements in that tree and you can also modify, delete and create leaves and branches by using the interfaces in the API.

3. SAX vs. DOM

The choice of SAX or the DOM is dependent on how much of a document the programmer wishes to access, ease of use, and performance concerns. The SAX treats a document as a series of events, which means it efficiently and swiftly, analyzes large XML documents. The drawback is that the programmer has to define the data structure to hold element data. The DOM must load the entire document in-memory before one has access. This takes more memory and time when compared to SAX. The DOM's strength is that the parser does almost everything, including reading the XML document in, creating a Java object model on top of it, and then gives a reference to this object model (a Document object) for manipulation. With the FIOM IDE speed and memory should not be an issue so the DOM's ease of use makes it the preferable choice to SAX.

E. TRANSLATIONS

Extensible Styles Language (XSL) is essentially three languages: a transformation language (XSLT), an accessing language (XPath), and a formatting language (XML-FO). Our research is only concerned with XSL Transformations (XSLT) and the XML Path language (XPath) used for pattern matching within XSLT style sheets.

XSLT is a high-level, declarative, and XML-based language. It allows a programmer to write XSLT code (style sheets) that transform an XML document into ***any*** text-based document. XSLT behaves as follows:

- First, an XSL engine is used to convert the XML document into a tree structure, which is composed of various types of nodes.
- Next, a style sheet is applied which transforms the XML document. The transformation is accomplished by using pattern matching, via XPath language notation, and then applying rules (templates) within the style sheet.
- In the transformed document, the body of the template element replaces the matched node in the source document.

XSLT does have a very limited, non-standardized, and awkward capability to do functional transformations by escaping into another language such as JavaScript or Java. The W3C's XSLT Recommendation does not define any aspect of this mechanism nor does it require that an XSLT processor should provide one at all. This should be remedied in future versions of XSLT. Until this is done, XSLT is not suitable for our purposes. Further discussion on extension functions can be found in [Kay00] on pages 122-124 and 632-637.

F. XML DATA-BINDING

1. An object-oriented view

The FIOM IDE uses XML Schema to represent the external interface of component systems being integrated. XML Schema allows us to define entities shared between systems. XML Schema also provides a mechanism for capturing information used to establish correspondence between two component system representations of the same real-world entity. The schema will define the structure, syntax, and to some extent, the semantics of the component system external interface. The OOMI methodology for resolving differences in view of real-world entities between component systems necessitates use of an object-orientated approach. If we view an XML Schema in object-oriented terms, a schema can be thought of as a class. Furthermore, an XML document, which is described and constrained by a particular schema, can also be thought of as an object. XML data binding is Java methodology, along with an API, allowing programs to be written that access and manipulate the content of XML documents in an object oriented fashion.

2. Why use data-binding?

The parse trees of the W3C DOM API and parser events of SAX API are primitive, constricting, and more focused on the structure vice the content of an XML document. Also, the DOM and SAX APIs treat all data as strings requiring the casting of data to a suitable type. In contrast, data binding facilitates the direct mapping (transforming) of an XML document to objects while maintaining the constraints imposed by its corresponding schema. Thus, all the benefits, power, and familiarity of the object-oriented paradigm are available. In effect, the programmer does not have to "reinvent the wheel" in gaining access to and updating the element content within a document. The programmer receives all of this along with the confidence that any resulting change in state will not violate well-formedness and validity of the resulting XML document.

3. Definition

The Java™ Architecture for XML Binding (JAXB) Working Draft Specification [Sun01] defines XML data-binding as a facility containing two components: A *schema compiler* and a *marshalling framework*. The *schema compiler* binds components of an input schema to derived lightweight classes. A lightweight class is conceptually the same as a *Java Bean* providing access to the content of the corresponding schema component via a set of accessor and mutator (*i.e.*, **get** and **set**) methods. The derived lightweight classes will maintain all the constraints described in its corresponding schema. This ensures that when the class instance (i.e., object) is unmarshalled it will not only be well-formed, but valid as well. The *marshalling framework* is a runtime API that, in conjunction with the derived lightweight classes, supports three primary operations:

- The *unmarshalling* of an XML document into a Java object that is an instance of a schema-derived class. This schema-derived class is composed of interrelated instances of both existing and schema-derived classes.
- The *marshalling* of an object back into an XML document.
- And, the *validation* of member variables against the constraints expressed in the schema.

In summary, the generated lightweight class will contain the following:

- Member variables representing the content of the input XML Schema.
- Get and set methods to access the generated member variables while maintaining constraints of the original schema.
- The unmarshal, marshal, and validate methods to convert an XML document into an instance object of the generated lightweight class and back.

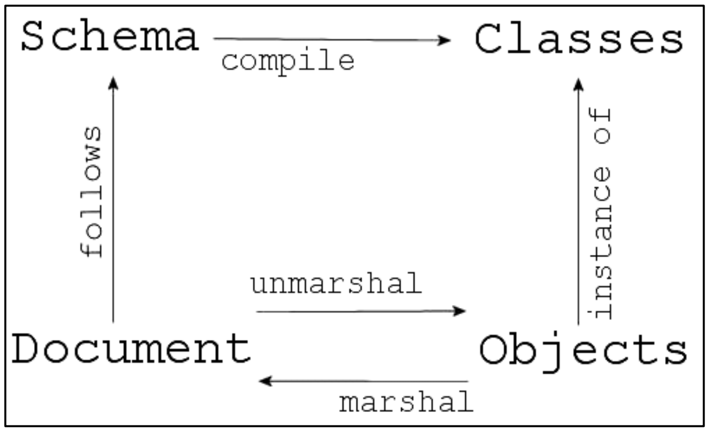


Figure III-2. XML and Java Relationships. From Ref. [Sun01].

Figure III-2 above shows how XML Schemas, XML documents, Java classes, and Java objects are related under this framework. As you can see, these relationships preserve equivalence, i.e., round tripping. In other words, the unmarshalling of an XML Document and then immediate marshalling of the Java object(s) produced should result in an equivalent copy of the original XML Document.

4. Implementations

This "bleeding-edge" technology is still in development. While, all current implementations support DTD schema, not one has yet to fully support all three parts of the 2 May 2001 W3C XML Schema recommendation. As discussed earlier, DTD specifications allow the validation of data structure, but do not allow the validation of data content, i.e., strong typing. This strong typing is necessary to allow for the syntactic and semantic matching in connecting different representations of the same real-world entity. There are currently three somewhat competitive APIs and one tool available that implement data binding: JAXB, Zeus, Castor, and Breeze XML Studio. We will discuss the merits of each and which implementation we chose to incorporate into our tool.

a. JAXB

First, Sun Microsystems' Java Architecture for Data Binding (JAXB) is being developed under the Java Community Process under Java Specification Request (JSR-031) [JSR31]. JAXB is essentially the base for the other three implementations and therefore the least developed. JAXB has an early access implementation available. This implementation is far from being a useable technology and currently only supports DTD schema. It also imposes some unreasonable restrictions on generated classes. For example, generated classes must extend some base classes from Sun's Java API for XML Processing (JAXP). It also generates classes that contain parsing code. This means that every change in your schema mapping requires code generation. So the only way to retain the code you wish to add to the generated JAXB code is to subclass their classes. This is probably a good idea regardless, but a constraint nonetheless. These constraints and the fact that JAXB is hardly a Beta make it unsuitable for use in our tool.

b. Zeus

Zeus is the open source brainchild of Brett McLaughlin, an Enhydra strategist with Lutris Technologies and one of the most knowledgeable individuals in the data-binding field. Much of McLaughlin's impetus in starting Zeus was his frustration with the complexity of the Castor project. Castor is more of a "Swiss-army knife" approach, trying to provide JDO (Java Data Objects), LDAP mappings, SQL mappings, etc., etc. This makes for a rather enormous code base, something McLaughlin was explicitly aiming to avoid in starting Zeus. Unlike Castor, Zeus provides a tightly focused XML to Java mapping in a very easy-to-use package. Zeus is quite a bit further along, in terms of usability, when compared with JAXB. Even so, it still only has minor support for XML Schema and none for data validation. [Zeus01]

c. Castor

Castor may be the "Swiss-army knife" of data binding API's but it's for good reason. It is one of the most active open-source projects we have witnessed. While all this individual input does adds to the complexity of the code, it does have the benefit of being constantly tested, evaluated, and improved. It also has XML Schema support along with data validation. This made Castor the clear winner for use in our tool. Currently the generated Java source files that Castor's Source Code Generator (schema compiler) produces need to be compiled. This is important if our tool is to use reflection in the viewing the structure of the generated classes. The Castor project will be adding code to the API eventually to handle this automatically [Cas01].

d. Breeze XML Studio

Breeze XML Studio Release 2.5 Beta II supports the XML Schema Recommendation of May 2, 2001 as well as support for XML Namespaces [Bre01]. Breeze Factor, LLC sits on the expert committee for JAXB JSR-031 and is committed to providing support if and when JSR-031 is useable [JSR31]. Even with these positive statements the negatives have far greater weight. The number one negative is that Breeze XML studio is proprietary, with no commercial competition whatsoever, and the company that owns it has the ability to radically affect the JSR-031. This is a very dangerous combination indeed. Their product is stand-alone with no capability to plug into another application. Also, although the Java classes generated are more compact and readable than the other tools, they are non-editable and have a birth certificate, i.e., a limited life that only Breeze LLC determines. This 2.5 Beta II was, for our project, a day late and a dollar short. We had already developed our own prototype that essentially has the same features of Breeze XML Studio using our own code and open source software.

G. ALTERNATIVES AND CONCLUSION

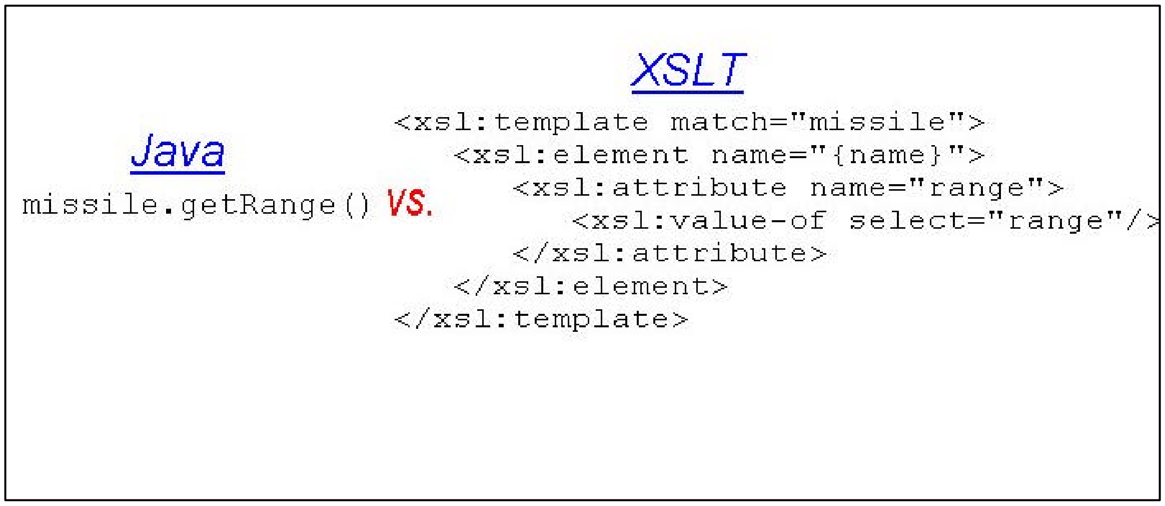
Can't we just use XSLT to generate Java classes? Yes, but given the choice, would you rather manually develop a style sheet for every document you need to manipulate in Java or have the code generated for you? With data binding you can take an arbitrary XML document and automatically generate the corresponding java classes. Likewise, who wants to tediously update a style sheet when there is a change in the schema? With data binding when a change is made in a schema, all that needs to be done is to re-generate the code in question. This automation dramatically reduces the time and effort to generate Java classes and thus reduces costs to develop and maintain code.

Why use data-binding at all? Can't we use SAX or DOM to access the document and XSLT to do the translations? The problems with this approach are as follows:

- Remember, with SAX the programmer has to design his own data model to handle a specific XML Document. With data-binding the mapping to a Java data model (class) is done automatically.

- DOM gives access to a document view as a tree structure. This involves tree searches to find the particular node a programmer is seeking. With data-binding all that is needed are simple get and set methods to access the data members of an object.

- XSLT can be used to transform an XML document to another view of that document. But, the problem is in scalability. Creating an XSLT style sheet is a manual process. This involves creating templates that match a certain element within an XML document and then describing what is to be done. This is very inefficient as can be seen in Figure III-3, which shows the effort required to retrieve the range of a missile in Java and in XSLT.



```
missile.getRange()
```

```
<xsl:template match="missile">
   <xsl:element name="{name}">
      <xsl:attribute name="range">
         <xsl:value-of select="range"/>
      </xsl:attribute>
   </xsl:element>
</xsl:template>
```

Figure III-3. Java vs. XSLT

Data-binding is the technology best suited to implement the FIOM IDE. It automatically maps an XML Schema into a Java class freeing the programmer from learning and using the low-level and tedious SAX and DOM API’s. It allows for functional translations using the power and vast resource libraries of the Java programming language directly. It also automatically moves XML documents to Java objects and back. There are no other technologies that give object-oriented access to

XML documents with no effort on the part of the programmer. This makes data-binding the clear winner.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. VISION, REQUIREMENTS, AND PROTOTYPE

A. INTRODUCTION

In this chapter we will briefly discuss the key aspects of the Vision Document, Software Requirements Specification (SRS), and prototype for the FIOM IDE. The Java source code and required files for the FIOM IDE can be found in appendix A. For the sake of brevity and because much of the information from the Vision Document and the SRS are already present in this thesis, they will not be included as appendices.

B. VISION DOCUMENT

The Vision document describes the application in general terms, including descriptions of the target market, the system users, and the application features [Leff00]. Again, the bulk of this information can be found in this thesis (see, Chapter 1). However, We would like to cover two important aspects of this document, the product features and the key use case.

1. Product Features

Table IV-1 below shows the current, but by no means exhaustive, product features for the FIOM IDE. These features provide a fundamental basis for product definition, scope management, and project management [Leff00].

ID	*Feature*	*Status*	*Priority*	*Effort*	*Assigned*	*Reason*
1	Access of component system schema files	Incorporated	Critical	Low	Christie	
2	View schema files in text only format	Incorporated	Important	Low	Christie	Used for text searches of schema
3	View schema files in graphical format	Incorporated	Critical	High	Christie	Help visualize structure of schema
4	Transform Component System Schema to Java Classes	Incorporated	Critical	High	Christie	Key to manipulating component RWE in Java
5	View FIOM graphically as hierarchy	Proposed	Critical	High	TBD	Help visualize structure of FIOM and manually find FEs quicker
6	View FE in text format	Proposed	Important	Med	TBD	
7	Search FIOM for	Proposed	Critical	High	Pugh	Core of IDE

ID	Feature	Status	Priority	Effort	Assigned	Reason
	FE that matches selected object in component system					
8	Display probability of match on returned FE	Proposed	Critical	Med	Pugh	Help IE choose most likely FE that matches object from component system schema while not filtering out low probability matches.
9	Split screen view of an object from the component system and a selected FE	Incorporated	Important	Low	Christie	Help IE determine if match a help with mapping of attributes
10	Mapping of attributes between component system object and FCR	Proposed	Critical	High	TBD	Choosing a transformation function will also be included
11	Manually add FE to FIOM	Proposed	Critical	Med	TBD	
12	Manually delete FE from FIOM	Proposed	Important	Low	TBD	
13	Edit FE	Proposed	Important	Med	TBD	
14	Print component system schema text format	Proposed	Useful	Low	TBD	
15	Print component system schema graphical format	Proposed	Useful	Med	TBD	
16	Print FE in text format	Proposed	Useful	Low	TBD	
17	Print FIOM hierarchy	Proposed	Useful	Med	TBD	
18	Save work as current "snapshot"	Incorporated	Useful	Med	Christie	Keep current view of files and FIOM to allow quick return to work.

Table IV-1. Product Features

2. Key Use Case

Table IV-2 below is the key use case for the FIOM IDE. Figure IV-1 shows the exemplary use cases and actors along with their relationships.

Item	Value
Use case name	High-level system view
Brief description	This use case describes how an interoperability engineer (IE) loads a component system schema and what the system state is

Item	*Value*
	after this operation is finished. This is the heart of what this tool is to provide.
Flow of events	1. IE identifies part of a component system to be integrated into the federation by selecting its XML schema file. The XML schema represents real world entities that are based upon the inputs, outputs, and services of a component system. 2. IE utilizes a software tool that analyzes and parses the XML schema. An XML data binding Java class generator automatically maps the XML schema to light-weight Java classes. This results in a "graph" of all the classes found in the component system schema. Each node of the graph represents member variable within the schema. A member variable may be a class itself. If this is true, the class node can be "clicked-on" and display its own member variables. 3. After selecting a class using the first software tool on the system, the IE uses a second software tool, which conducts a search to find the FIOM representation of the same real world entity. If no matches are found, IE is given option to create a new Federation Entity (FE). Else, a list of FE with a probability of match for each is displayed. 4. The IE analyzes the output from Step 3, and determines which member variables are appropriately mapped. The tool helps this process by displaying both graphs of the component system object and the FCR side by side. If the object does not map completely, the IE is given the option to add the current component system object to the FIOM hierarchy as a new FCR. This new FCR may inherit the FCR that was the closest match from the search. 5. The IE executes a fourth tool that sets the functional transformations needed to map the component system object to the correct FCR and vice versa
Exceptional flow of events	TBD.
Preconditions	A well-formed XML schema representing a real world entity to be integrated into the federation is available.

Item	***Value***
Postconditions	The FIOM is updated with the addition of the XML schema from the component system in question. The FIOM will now be capable of taking an XML Document based upon the XML schema previously integrated and transform it into another XML Document based upon another integrated component system's XML schema. This is provided that the XML schemas represent the same real world object.
Special requirements	None.

Table IV-2. Key Use Case

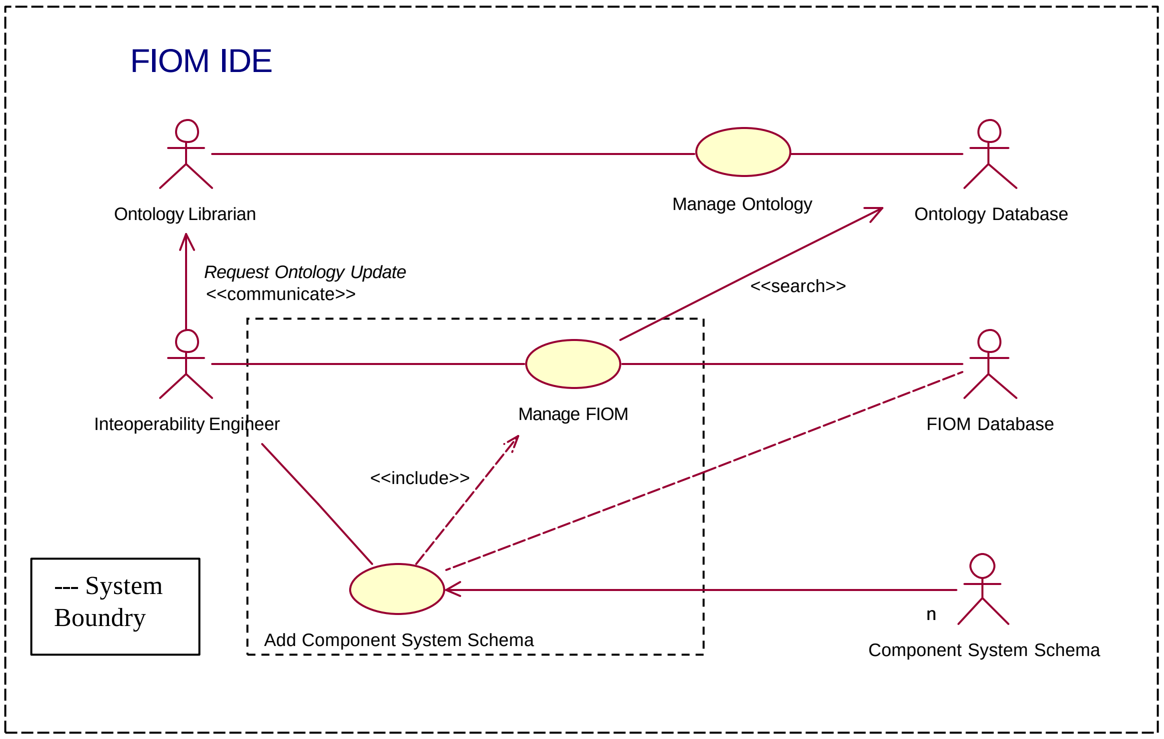


Figure IV-1. Key Use Case Diagram

C. SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

The highlights of the SRS covered in this section are the Use Case Model Survey, Actor Survey, and the Activity Diagrams describing the key use case in Table IV-2.

1. Use Case Model Survey

Name	*Description*	*Actor(s)*
Add Component System Schema	Allows IE to open a component system XML Schema file and display the objects contained therein. Using a data-binding mechanism, the file is transformed into a light-weight Java class(es) . A search is then conducted to find a matching FCR within the FIOM. The results of the match are displayed. A FCR is chosen, or one is created (see Manage FIOM), and the relationship is made. The translations between member variables are made. Lastly, the relationship and translation(s) are stored in the FIOM.	IE, Component System Schema, FIOM Database
Manage FIOM	Gives the IE the ability to create, modify, or delete an FE, FEV view, FCR, or CCR. Note: CCRs will not be modified.	IE, FIOM Database, Ontology Database
Manage Ontology	Adds a new entity to the Ontology Database. Not part of FIOM IDE.	Ontology Librarian Ontology Database

Table IV-3. Use Case Model Survey

2. Actor Survey

Actor Name	*Description*
Component System External Interface Schema	XML schema file defining the component system's external interface and containing a representation of the real-world entity it wishes to add to the FIOM.
FIOM Database	The collection of Federation Entities (FE) containing the relationships and translations necessary for component system members to share information the members have registered using the FIOM IDE.

Actor Name	Description
Interoperability Engineer (IE)	An individual that is an experienced Software Engineer that is responsible for integrating DoD systems. It is assumed that the IE is not necessarily a domain expert in the particular systems that are to be integrated. But, the IE is expert in the federation namespace being used to create the FIOM. Once the FIOM is created it will be used within a network to convert messages and services between component systems.
Ontology Librarian	One who has the ability and authority to update the contents of the Ontology Database.
Ontology Database	The working model of entities and interactions in some particular domain of knowledge or practices, such as Defense Information Systems Agency's (DISA's) Defense Information Infrastructure Common Operating Environment (DII COE) XML Registry and the Defense Modeling and Simulation Office's (DMSO's) Functional Description of the Mission Space (FDMS) namespaces [DII01, FDM01]. In artificial intelligence (AI), an ontology is, according to Tom Gruber, an AI specialist at Stanford University, "the specification of conceptualizations, used to help programs and humans share knowledge." In this usage, an ontology is a set of concepts - such as things, events, and relations - that are specified in some way (such as specific natural language) in order to create an agreed-upon vocabulary for exchanging information [Whatis].

Table IV-4. Actor Survey

3. Activity Diagrams

Figures IV-2 and IV-3 describe the steps needed to implement the key use case found in Table IV-3.

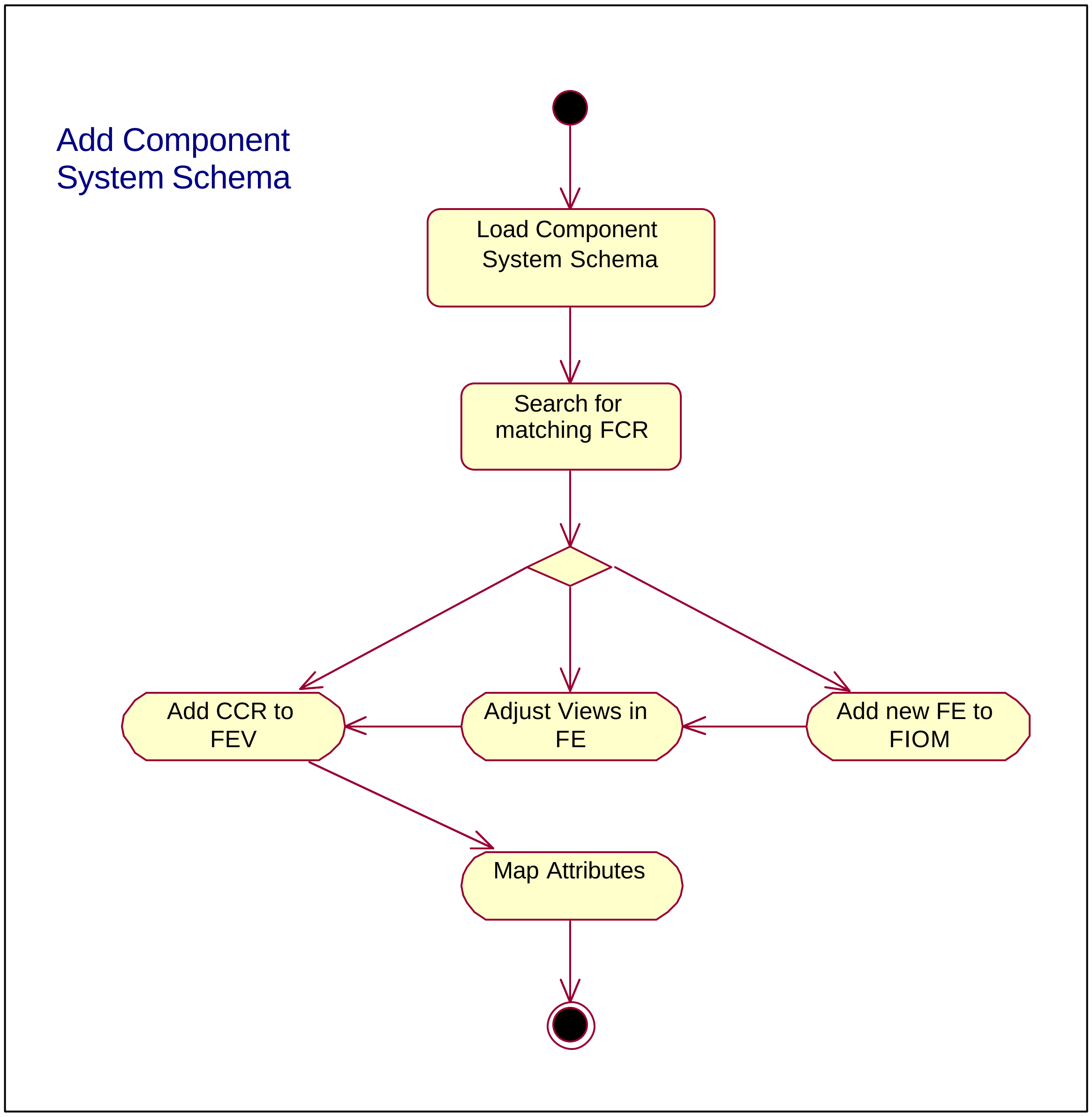


Figure IV-2. Add Component System Schema Activity Diagram

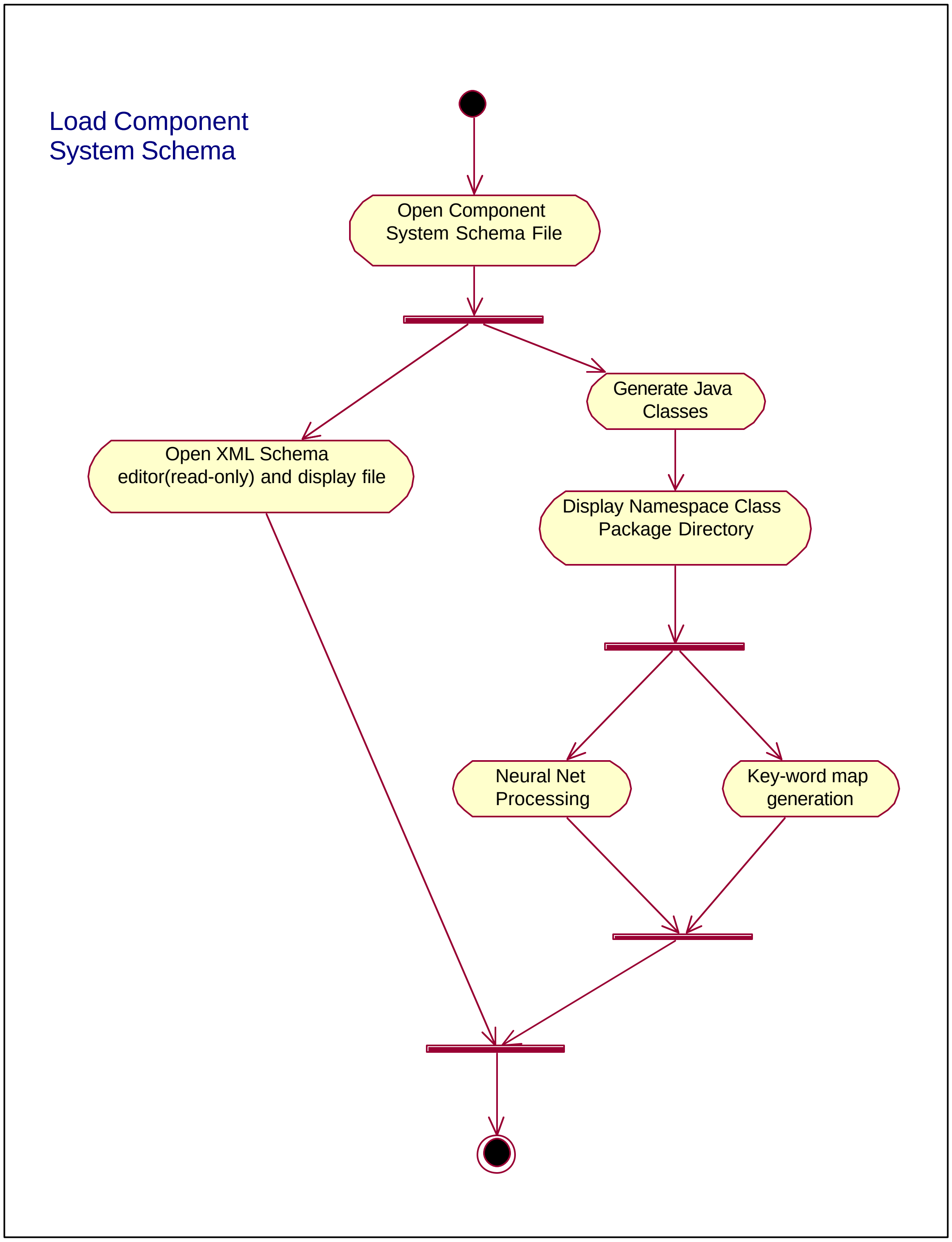

Load Component
System Schema
Open Component System Schema File
Generate Java Classes
Open XML Schema editor(read-only) and display file
Display Namespace Class Package Directory
Neural Net Processing
Key-word map generation

Figure IV-3. Load Component System Schema Activity Diagram

D. FIOM IDE PROTOTYPE

The main goal of our work with the prototype was to build the base graphical user environment and the data binding mechanism needed for follow-on work. In this section we will to discuss the current features implemented and cover an example of exercising the data-binding feature of the application.

1. Environment

The FIOM IDE will execute on a system running Java 2 runtime environment 1.3.0 with normal 800x600 resolution or greater monitor, keyboard, and mouse. All user interface components are implemented using the Java Swing library. Upon program start an 800x600 pixel window will open and be placed in the middle of the screen. The application will have the "Look and Feel" of the operating system environment it is running in. Figure IV-6 shows a typical window running under a Windows 2000 operating system environment. Anyone familiar with any Windows type environment should find this application to be "user friendly."

2. Functionality

There are four drop-down menus used to control the FIOM IDE. Table IV-5 describes their functions and submenus. These drop-down menus can be seen in Figure IV-4.

Menu	*Sub-Menu*	*Function*
File		Basic File Input/Output operations
	Load XML Schema File	JFileChooser is displayed to allow user to load .xsd file into tool. Figure IV-5 shows a sample dialog box.
	Close XML Schema File	Current schema file is closed along with associated view XML Schema text or tree panes opened.
	Exit	Closes all files and exits the application
View		Displays different views of schema file loaded.

Menu	*Sub-Menu*	*Function*
	XML Schema Tree	As seen in Figure IV-6, displays a JTree containing the contents of the current schema file. This tree also contains any <include> files within the original schema file as well as the <include> files themselves.
	XML Schema Text	As seen in Figure IV-7, XHTML representation of the original schema file only. <include> files are not added.
Tools		Handles different tools the application has.
	Generate Java Classes	Takes the loaded XML Schema file and creates lightweight Java classes and stores them in the directory based on the package given by the user. The package dialog can be seen in Figure IV-8.
	Options	Displays application options and allows the user to edit them. Currently path information for schema location and FIOM location can be edited and stored. Figure IV-9 shows an example of the dialog. The information is stored in a XML Document called options.xml. **Note: This tool actually helped in building itself!** This operation was created using the FIOM IDE. The options XML Schema was created and then loaded into the FIOM IDE. The classes were then generated and then incorporated into the FIOM IDE code. This gave the FIOM IDE the ability to manipulate an options XML file.

Menu	***Sub-Menu***	***Function***
Help		Handles any help or information screens.
	About	Gives brief background on tool as seen in Figure IV-10.

Table IV-5. Menu Functions

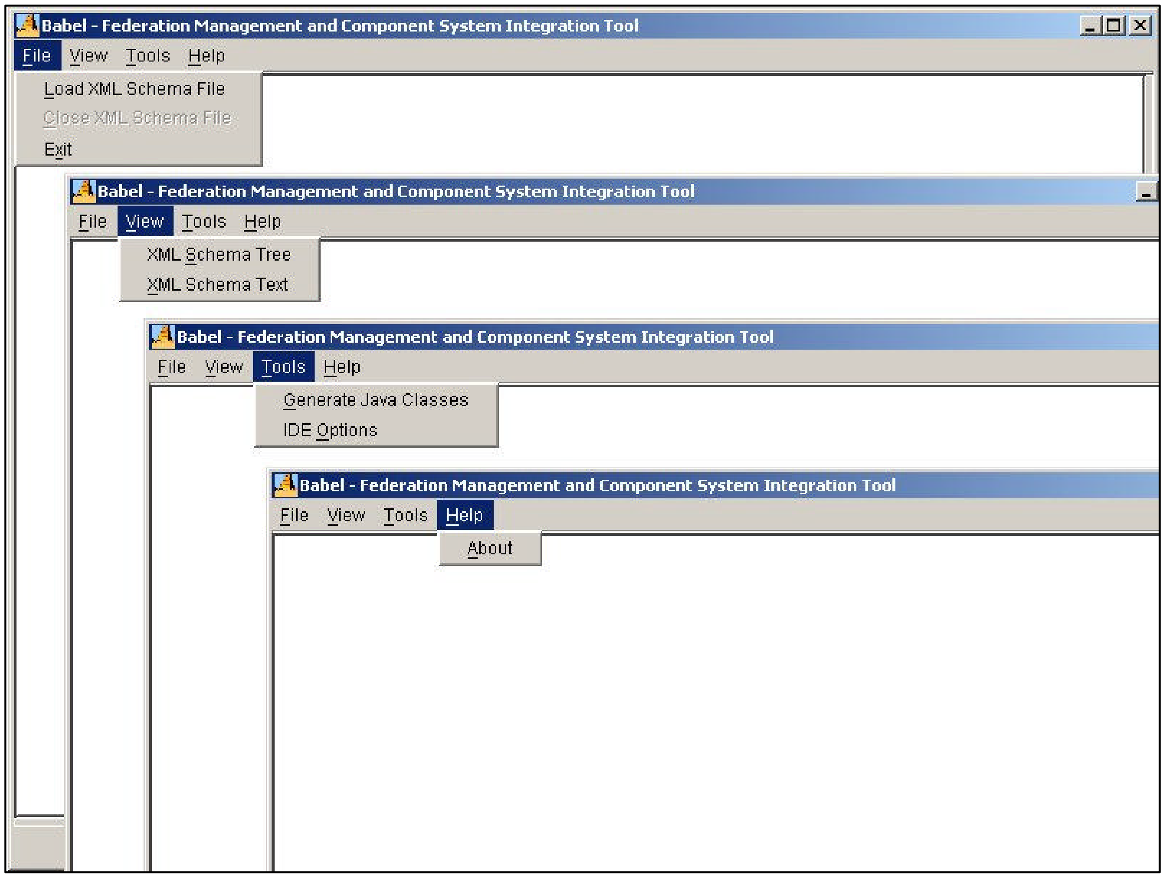


Figure IV-4. Cascaded View of Menus

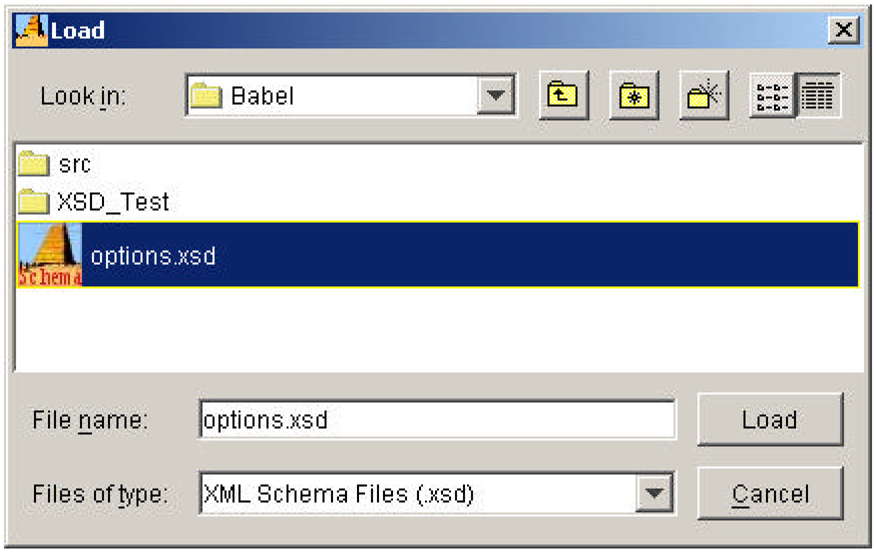
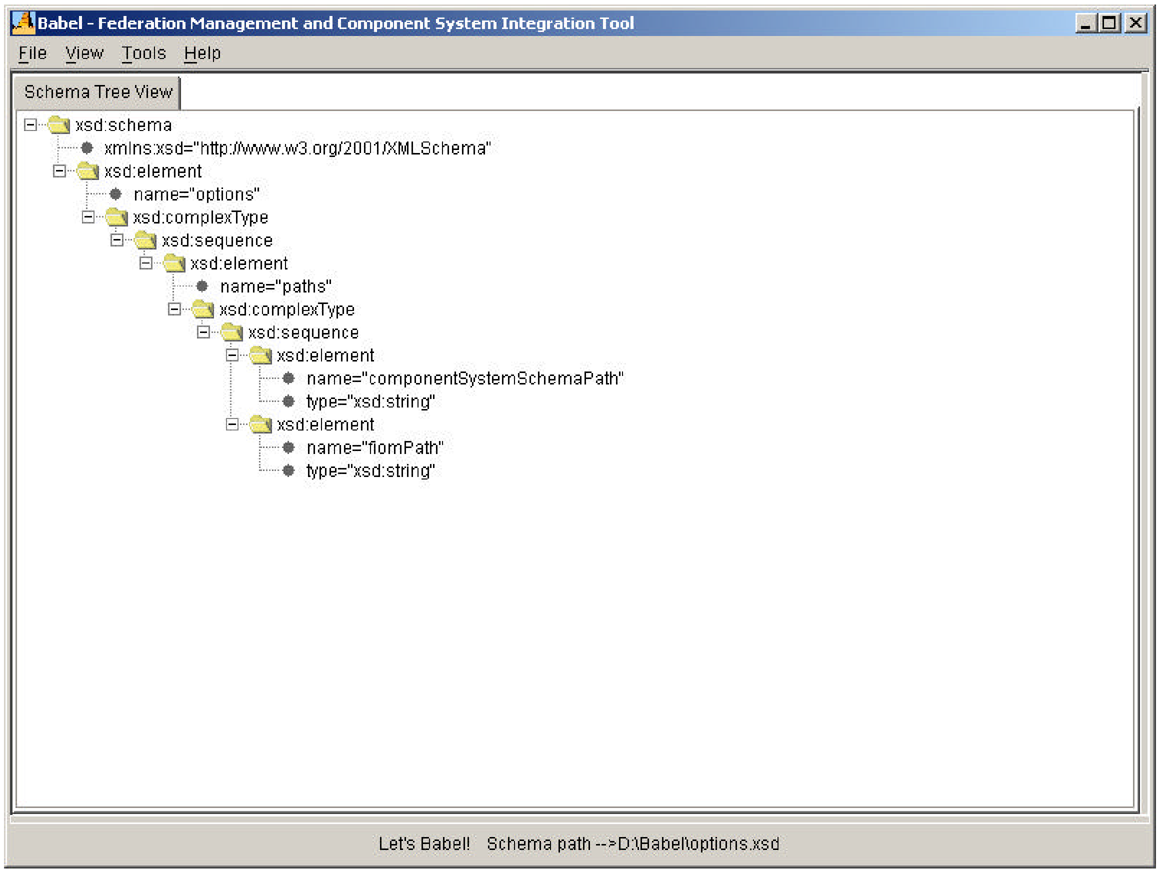


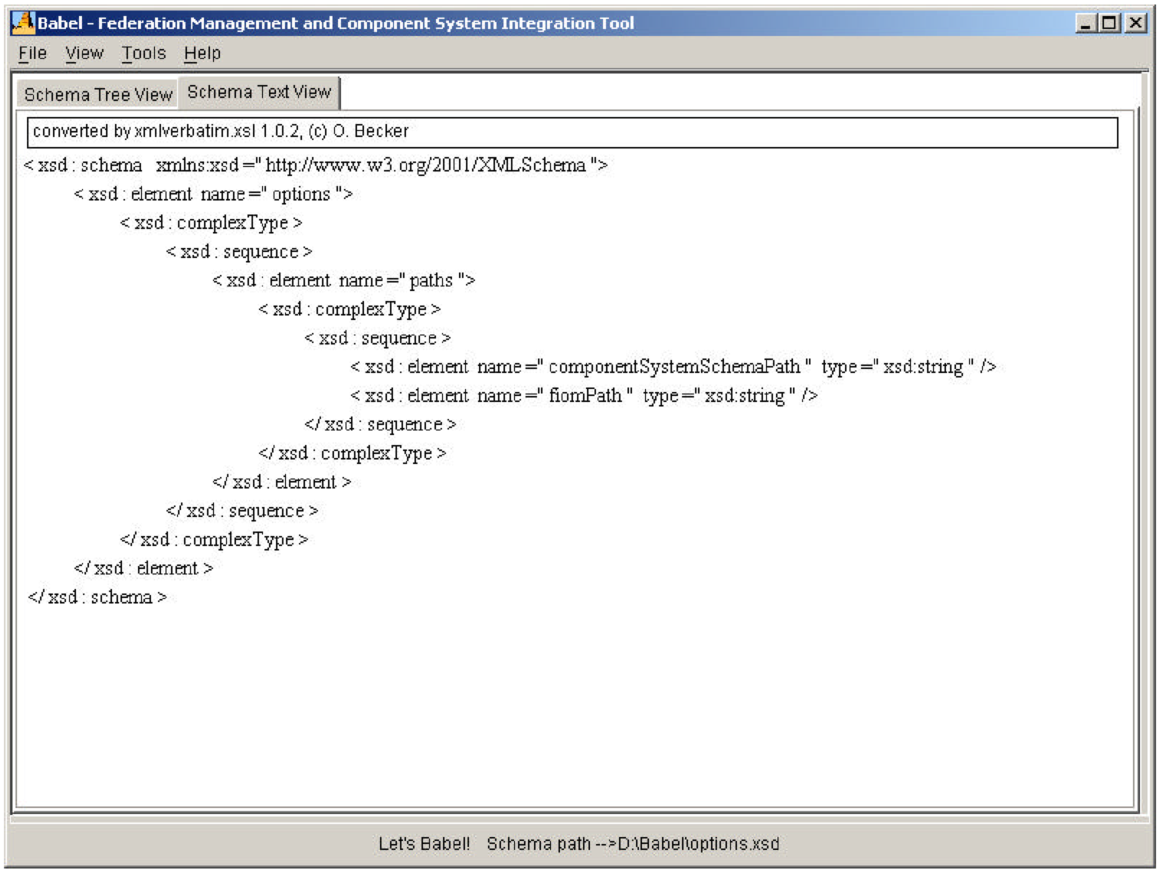
Figure IV-5. Load Schema Dialog


Babel - Federation Management and Component System Integration Tool
File View Tools Help
Schema Tree View

```
xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsd:element
    name="options"
    xsd:complexType
      xsd:sequence
        xsd:element
          name="paths"
          xsd:complexType
            xsd:sequence
              xsd:element
                name="componentSystemSchemaPath"
                type="xsd:string"
              xsd:element
                name="fomPath"
                type="xsd:string"
```

Let's Babel!   Schema path -->D:\Babel\options.xsd

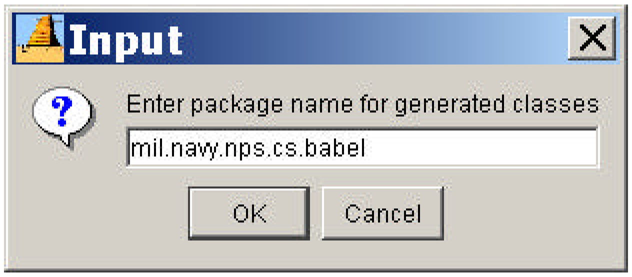
Figure IV-6. Schema Tree View Display


Babel - Federation Management and Component System Integration Tool
File View Tools Help
Schema Tree View
Schema Text View

```
converted by xmlverbatim.xsl 1.0.2, (c) O. Becker
< xsd : schema  xmlns:xsd =" http://www.w3.org/2001/XMLSchema ">
    < xsd : element  name =" options ">
        < xsd : complexType >
            < xsd : sequence >
                < xsd : element  name =" paths ">
                    < xsd : complexType >
                        < xsd : sequence >
                            < xsd : element  name =" componentSystemSchemaPath "  type =" xsd:string " />
                            < xsd : element  name =" fiomPath "  type =" xsd:string " />
                        </ xsd : sequence >
                    </ xsd : complexType >
                </ xsd : element >
            </ xsd : sequence >
        </ xsd : complexType >
    </ xsd : element >
</ xsd : schema >
```

Let's Babel!   Schema path -->D:\Babel\options.xsd

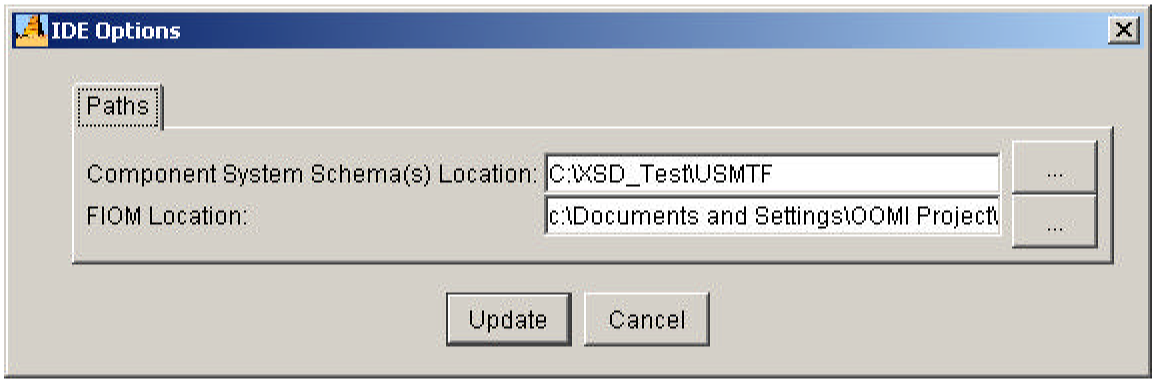
Figure IV-7.   Schema Text View Display



Enter package name for generated classes

mil.navy.nps.cs.babel

Figure IV-8.   Package Dialog for Generating Classes



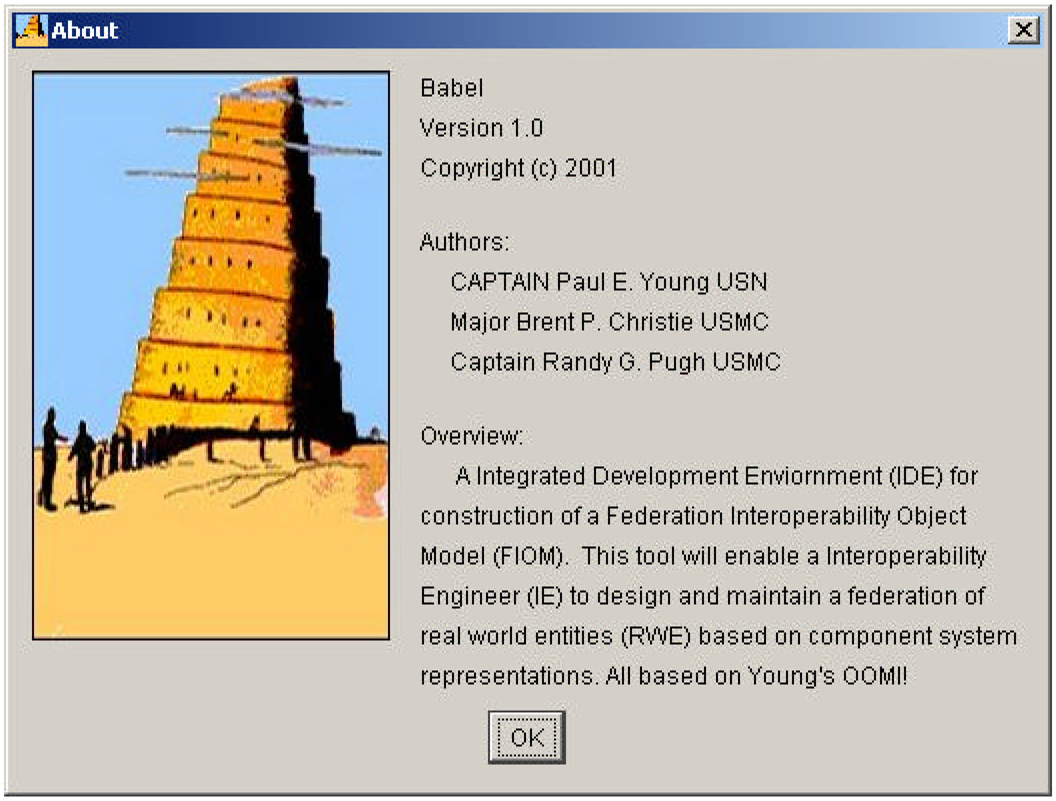
IDE Options

Paths

Component System Schema(s) Location: C:\XSD_Test\USMTF

FIOM Location: c:\Documents and Settings\OOMI Project\

Figure IV-9. IDE Options Dialog



About

Babel
Version 1.0
Copyright (c) 2001

Authors:
CAPTAIN Paul E. Young USN
Major Brent P. Christie USMC
Captain Randy G. Pugh USMC

Overview:
A Integrated Development Enviornment (IDE) for construction of a Federation Interoperability Object Model (FIOM). This tool will enable a Interoperability Engineer (IE) to design and maintain a federation of real world entities (RWE) based on component system representations. All based on Young's OOMI!

Figure IV-10. About Information Display

3. Example of Data-binding using the FIOM IDE

In this section we hope to demonstrate the power of data-binding by explaining how the FIOM IDE actually helped in building itself. Feature 18 in Table IV-1 sought to save work as a current "snapshot" in order to keep a current view of the component system files and the FIOM to allow a quick return to work. Part of capturing the current state of the application was the persistent storage and retrieval of path information for component system XML Schema and FIOM files. To incorporate this feature the following steps were taken:

- An XML Schema file called options.xsd, Figure IV-11, was developed to define the logical structure of the document used to store user options for the application. As can be seen below in Figure IV-11, XML documents following this schema will have a root element called "options", which has a sequence of children. Currently, the "options" element has only one child called "paths" also having a sequence of children called "componentSystemSchemaPath" and "fiomPath", both of which are strings. This schema can be easily modified to add more options that require persistent storage.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="options">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="paths">
               <xsd:complexType>
                  <xsd:sequence>
                     <xsd:element name="componentSystemSchemaPath"
type="xsd:string"/>
                     <xsd:element name="fiomPath" type="xsd:string"/>
                  </xsd:sequence>
               </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Figure IV-11. options.xsd

- The prototype FIOM IDE was then started and the options.xsd was loaded by clicking on the File menu and then the Load XML Schema File sub-menu.

- A JFileChooser dialog box then appears and the options.xsd file is located and chosen as seen in Figure IV-5.

- The file's content was then checked using the Schema Tree View and Schema Text View, which are sub-menus under the View menu. See Figure IV-6 and Figure IV-7.

- The Generate Java Classes sub-menu, which can be found under the Tools menu, was then clicked on and the Package Dialog Box appeared. The package for the FIOM IDE was then entered. Figure IV-8.

- Then FIOM generated four Java classes in background using the Castor Source Generator API. The four files are:
    - Options.java
    - OptionsDescriptor.java
    - Paths.java
    - And, PathsDescriptor.java

- These generated files can be seen in Appendix A. The attributes and methods contained in Options.java and Paths.java are listed in Table IV-6 below.

Class file	Attributes	Methods

Options.java	`Paths _paths`	`public Paths getPaths()`
		`public void setPaths(Paths paths)`
		`public boolean isValid()`
		`public void marshal(java.io.Writer out)`
		`public static Options unmarshal(java.io.Reader reader)`

Paths.java	`String _componentSystemSchemaPath`
	`String _fiomPath`
		`public String getComponentSystemSchemaPath()`
		`public void setComponentSystemSchemaPath(String`

Class file	Attributes	Methods
		componentSystemSchemaPath) public void setFiomPath(String fiomPath) public java.lang.String getFiomPath() public void setFiomPath(String fiomPath) public boolean isValid() public void marshal(java.io.Writer out) public static Paths unmarshal(java.io.Reader reader)

Table IV-6. Generated Class Files

- OptionsDescriptor and PathsDescriptor class serve a class reflection mechanism to retrieve things such as class name, its super class, namespace, etc.

- The methods in Table IV-6 are then used within the program to access the options.xml file, see Figure III-1. This programmatic access can be seen in the IDEOptions class constructor and display method in Appendix A.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION

A. RESULTS OF THIS RESEARCH

This research effort has resulted in the beginnings a FIOM IDE prototype capable of demonstrating the *transformation* process. This was originally not the focus of our research. But, it became clear as work progressed, that a way to view real-world entities in pure object-orientated terms was needed if the project was to move forward.

XML, and the family of technologies associated with it, are very immature at this time. Most of the recommendations from the W3C are less than two years old and some are still only candidates. We found that the XML Schema inheritance mechanism was too crude and XSLT's translation capabilities were insufficient to effect proper functional translations. Our frustration grew as we tried to make XML "fit" in our object-orientated model. Luckily, we stumbled across one of the newest XML technologies, data-binding. Data-binding enabled the *transformation* of XML Schemas into pure Java classes, and thus, freed us from trying to implement object-orientated techniques using other crude, immature, and sometimes unusable XML technologies.

B. CRITICISMS OF THIS RESEARCH

The new capabilities that XML brings to interoperation between legacy systems are a two-edged sword. XML now allows the world to define the structure and semantics of data to suit its needs. But again, XML technologies are very immature and it will be quite some time before data-binding, or any other XML technology, performs as advertised.

C. RECOMMENDATIONS

Follow on work should focus on the following:

- The classes generated via data-binding need to be compiled if we are to use reflection to view the contents of the classes. The Castor API will eventually do this automatically but, until then, a manual solution is needed.

- Pugh's correlation techniques, see [Pug01], drew their discriminators from XML Schema files only. Since the federation view of a real-world entity is now described using Java Class only, his techniques must be modified.
- Lastly, a graphical, UML type, mapping interface is need to aid the Interoperability Engineer in developing the translation methods necessary to convert FCRs to CCRs, and visa versa. The University of California at Irvine developed an open source Java library called the Graph Editing Framework (GEF) that may drastically reduce the amount of work necessary to accomplish this task [GEF01].

APPENDIX A. FIOM IDE SOURCE CODE AND REQUIRED FILES

A. SOURCE CODE

1. Babel.java

```
package mil.navy.nps.cs.babel;

import javax.swing.UIManager;
import java.awt.*;

/**
 * Title:        Babel
 * Description:  A Integrated Development Enviornment (IDE) for construction of
 *               a Federation Interoperability Object Model (FIOM).  This tool
 *               will enable a Interoperability Engineer (IE) to design and
 *               maintain a federation of real world entities (RWE) based on
 *               component system representations.
 * Copyright:    Copyright (c) 2001
 * Company:      Naval Postgraduate School
 * @author Major Brent P. Christie USMC
 * @version 1.0
 */

public class Babel {

   // ***** DATA MEMBERS *****

   boolean packFrame = false;


   // ***** CONSTUCTORS *****

   /**Construct the application*/
   public Babel() {
      MainFrame frame = new MainFrame();
      //Validate frames that have preset sizes
      //Pack frames that have useful preferred size info, e.g. from their layout
      if ( packFrame ) {
         frame.pack();
      }
      else {
         frame.validate();
      }

      //Center the window
      Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
      Dimension frameSize = frame.getSize();
      if ( frameSize.height > screenSize.height ) {
         frameSize.height = screenSize.height;
      }
      if ( frameSize.width > screenSize.width ) {
         frameSize.width = screenSize.width;
      }
      frame.setLocation( ( screenSize.width - frameSize.width ) / 2,
                         ( screenSize.height - frameSize.height ) / 2 );
      frame.setVisible( true );
   }//end Babel class constructor


   // ***** METHODS *****

   /**Main method*/
   public static void main( String[] args ) {
      try {
         //UIManager.LookAndFeelInfo looks[];
         //looks = UIManager.getInstalledLookAndFeels();
         //0 - Metal, 1 - Motif, 2 - Windows
         //UIManager.setLookAndFeel( looks[0].getClassName() );
         UIManager.setLookAndFeel( UIManager.getSystemLookAndFeelClassName() );
      }
      catch( Exception e ) {
         e.printStackTrace();
      }
      new Babel();
   }//end of main method

}//end of class Bable

//eof Babel.java
```

2. ConvenientFileFilter.java

```
/*
 * @(#)ExampleFileFilter.java 1.9 99/04/23
 *
 * Copyright (c) 1998, 1999 by Sun Microsystems, Inc. All Rights Reserved.
 *
 * Sun grants you ("Licensee") a non-exclusive, royalty free, license to use,
 * modify and redistribute this software in source and binary code form,
 * provided that i) this copyright notice and license appear on all copies of
 * the software; and ii) Licensee does not utilize the software in a manner
 * which is disparaging to Sun.
 *
 * This software is provided "AS IS," without a warranty of any kind. ALL
 * EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY
 * IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR
 * NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND ITS LICENSORS SHALL NOT BE
 * LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING
 * OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS
 * LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT,
 * INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER
 * CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF
 * OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 *
 * This software is not designed or intended for use in on-line control of
 * aircraft, air traffic, aircraft navigation or aircraft communications; or in
 * the design, construction, operation or maintenance of any nuclear
 * facility. Licensee represents and warrants that it will not use or
 * redistribute the Software for such purposes.
 */

package mil.navy.nps.cs.babel;

import java.io.File;
import java.util.Hashtable;
import java.util.Enumeration;
import javax.swing.*;
import javax.swing.filechooser.*;

/**
 * A convenience implementation of FileFilter that filters out
 * all files except for those type extensions that it knows about.
 *
 * Extensions are of the type ".foo", which is typically found on
 * Windows and Unix boxes, but not on Macinthosh. Case is ignored.
 *
 * Example - create a new filter that filerts out all files
 * but gif and jpg image files:
 *
 *     JFileChooser chooser = new JFileChooser();
 *     ExampleFileFilter filter = new ExampleFileFilter(
 *                   new String{"gif", "jpg"}, "JPEG & GIF Images")
 *     chooser.addChoosableFileFilter(filter);
 *     chooser.showOpenDialog(this);
 *
 * @version 1.9 04/23/99
 * @author Jeff Dinkins
 */
public class ConvenientFileFilter extends FileFilter {

    private static String TYPE_UNKNOWN = "Type Unknown";
    private static String HIDDEN_FILE = "Hidden File";

    private Hashtable filters = null;
    private String description = null;
    private String fullDescription = null;
    private boolean useExtensionsInDescription = true;

    /**
     * Creates a file filter. If no filters are added, then all
     * files are accepted.
     *
     * @see #addExtension
     */
    public ConvenientFileFilter() {
          this.filters = new Hashtable();
    }

    /**
     * Creates a file filter that accepts files with the given extension.
     * Example: new ExampleFileFilter("jpg");
     *
     * @see #addExtension
     */
    public ConvenientFileFilter(String extension) {
          this(extension,null);
    }

    /**
     * Creates a file filter that accepts the given file type.
     * Example: new ExampleFileFilter("jpg", "JPEG Image Images");
     *
     * Note that the "." before the extension is not needed. If
     * provided, it will be ignored.
     *
     * @see #addExtension
     */
    public ConvenientFileFilter(String extension, String description) {
          this();
          if(extension!=null) addExtension(extension);
          if(description!=null) setDescription(description);
    }

    /**
     * Creates a file filter from the given string array.
     * Example: new ExampleFileFilter(String {"gif", "jpg"});
     *
     * Note that the "." before the extension is not needed adn
     * will be ignored.
     *
     * @see #addExtension
     */
    public ConvenientFileFilter(String[] filters) {
          this(filters, null);
    }

    /**
     * Creates a file filter from the given string array and description.
     * Example: new ExampleFileFilter(String {"gif", "jpg"}, "Gif and
     *                                                           JPG Images");
     *
     * Note that the "." before the extension is not needed and will be ignored.
     *
     * @see #addExtension
     */
    public ConvenientFileFilter(String[] filters, String description) {
           this();
           for (int i = 0; i < filters.length; i++) {
              // add filters one by one
              addExtension(filters[i]);
           }
           if(description!=null) setDescription(description);
    }

    /**
     * Return true if this file should be shown in the directory pane,
     * false if it shouldn't.
     *
     * Files that begin with "." are ignored.
     *
     * @see #getExtension
     * @see FileFilter#accepts
     */
    public boolean accept(File f) {
           if(f != null) {
              if(f.isDirectory()) {
                     return true;
         }
         String extension = getExtension(f);
         if(extension != null && filters.get(getExtension(f)) != null) {
                     return true;
         };
      }
           return false;
    }

    /**
     * Return the extension portion of the file's name .
     *
     * @see #getExtension
     * @see FileFilter#accept
     */
     public String getExtension(File f) {
      if(f != null) {
         String filename = f.getName();
         int i = filename.lastIndexOf('.');
         if(i>0 && i<filename.length()-1) {
            return filename.substring(i+1).toLowerCase();
         };
      }
      return null;
    }

    /**
     * Adds a filetype "dot" extension to filter against.
     *
     * For example: the following code will create a filter that filters
     * out all files except those that end in ".jpg" and ".tif":
     *
     *   ExampleFileFilter filter = new ExampleFileFilter();
     *   filter.addExtension("jpg");
     *   filter.addExtension("tif");
     *
     * Note that the "." before the extension is not needed and will be ignored.
     */
    public void addExtension(String extension) {
           if(filters == null) {
              filters = new Hashtable(5);
           }
           filters.put(extension.toLowerCase(), this);
           fullDescription = null;
    }


    /**
```

```
             * Returns the human readable description of this filter. For
             * example: "JPEG and GIF Image Files (*.jpg, *.gif)"
             *
             * @see setDescription
             * @see setExtensionListInDescription
             * @see isExtensionListInDescription
             * @see FileFilter#getDescription
             */
            public String getDescription() {
                  if(fullDescription == null) {
                     if(description == null || isExtensionListInDescription()) {
                             fullDescription = description==null ? "(" : description + "
(";
                             // build the description from the extension list
                             Enumeration extensions = filters.keys();
                             if(extensions != null) {
                                fullDescription += "." + (String)
extensions.nextElement();
                                while (extensions.hasMoreElements()) {
                                        fullDescription += ", " + (String)
extensions.nextElement();
                                }
                             }
                             fullDescription += ")";
                 }
                 else {
                             fullDescription = description;
                 }
                  }
                  return fullDescription;
            }

            /**
             * Sets the human readable description of this filter. For
             * example: filter.setDescription("Gif and JPG Images");
             *
             * @see setDescription
             * @see setExtensionListInDescription
             * @see isExtensionListInDescription
             */
            public void setDescription(String description) {
                  this.description = description;
                  fullDescription = null;
            }

            /**
             * Determines whether the extension list (.jpg, .gif, etc) should
             * show up in the human readable description.
             *
             * Only relevent if a description was provided in the constructor
             * or using setDescription();
             *
             * @see getDescription
             * @see setDescription
             * @see isExtensionListInDescription
             */
            public void setExtensionListInDescription(boolean b) {
                  useExtensionsInDescription = b;
                  fullDescription = null;
            }

            /**
             * Returns whether the extension list (.jpg, .gif, etc) should
             * show up in the human readable description.
             *
             * Only relevent if a description was provided in the constructor
             * or using setDescription();
             *
             * @see getDescription
             * @see setDescription
             * @see setExtensionListInDescription
```

```
     */
    public boolean isExtensionListInDescription() {
          return useExtensionsInDescription;
    }

}//end ConvenientFileFilter class

//eof ConvenientFileFilter.java
```

3. ConvenientFileView.java

```
/*
 * @(#)ExampleFileView.java   1.8 99/04/23
 *
 * Copyright (c) 1998, 1999 by Sun Microsystems, Inc. All Rights Reserved.
 *
 * Sun grants you ("Licensee") a non-exclusive, royalty free, license to use,
 * modify and redistribute this software in source and binary code form,
 * provided that i) this copyright notice and license appear on all copies of
 * the software; and ii) Licensee does not utilize the software in a manner
 * which is disparaging to Sun.
 *
 * This software is provided "AS IS," without a warranty of any kind. ALL
 * EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY
 * IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR
 * NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND ITS LICENSORS SHALL NOT BE
 * LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING
 * OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS
 * LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT,
 * INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER
 * CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF
 * OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 *
 * This software is not designed or intended for use in on-line control of
 * aircraft, air traffic, aircraft navigation or aircraft communications; or in
 * the design, construction, operation or maintenance of any nuclear
 * facility. Licensee represents and warrants that it will not use or
 * redistribute the Software for such purposes.
 */

package mil.navy.nps.cs.babel;

import javax.swing.*;
import javax.swing.filechooser.*;

import java.io.File;
import java.util.Hashtable;

/**
 * A convenience implementation of the FileView interface that
 * manages name, icon, traversable, and file type information.
 *
 * This this implemention will work well with file systems that use
 * "dot" extensions to indicate file type. For example: "picture.gif"
 * as a gif image.
 *
 * If the java.io.File ever contains some of this information, such as
 * file type, icon, and hidden file inforation, this implementation may
 * become obsolete. At minimum, it should be rewritten at that time to
 * use any new type information provided by java.io.File
 *
 * Example:
 *    JFileChooser chooser = new JFileChooser();
 *    fileView = new ExampleFileView();
 *    fileView.putIcon("jpg", new ImageIcon("images/jpgIcon.jpg"));
 *    fileView.putIcon("gif", new ImageIcon("images/gifIcon.gif"));
 *    chooser.setFileView(fileView);
 *
 * @version 1.8 04/23/99
 * @author Jeff Dinkins
 */
public class ConvenientFileView extends FileView {

    private Hashtable icons = new Hashtable(5);
    private Hashtable fileDescriptions = new Hashtable(5);
    private Hashtable typeDescriptions = new Hashtable(5);

    /**
     * The name of the file.  Do nothing special here. Let
     * the system file view handle this.
     * @see #setName
     * @see FileView#getName
     */
    public String getName(File f) {
          return null;
    }

    /**
     * Adds a human readable description of the file.
     */
    public void putDescription(File f, String fileDescription) {
          fileDescriptions.put(fileDescription, f);
    }

    /**
     * A human readable description of the file.
     *
     * @see FileView#getDescription
     */
    public String getDescription(File f) {
          return (String) fileDescriptions.get(f);
    };

    /**
     * Adds a human readable type description for files. Based on "dot"
     * extension strings, e.g: ".gif". Case is ignored.
     */
    public void putTypeDescription(String extension, String typeDescription) {
          typeDescriptions.put(typeDescription, extension);
    }

    /**
     * Adds a human readable type description for files of the type of
     * the passed in file. Based on "dot" extension strings, e.g: ".gif".
     * Case is ignored.
     */
    public void putTypeDescription(File f, String typeDescription) {
          putTypeDescription(getExtension(f), typeDescription);
    }

    /**
     * A human readable description of the type of the file.
     *
     * @see FileView#getTypeDescription
     */
    public String getTypeDescription(File f) {
          return (String) typeDescriptions.get(getExtension(f));
    }

    /**
     * Conveinience method that returnsa the "dot" extension for the
     * given file.
     */
    public String getExtension(File f) {
          String name = f.getName();
          if(name != null) {
             int extensionIndex = name.lastIndexOf('.');
             if(extensionIndex < 0) {
                    return null;
```

```
            }
            return name.substring(extensionIndex+1).toLowerCase();
        }
        return null;
    }

    /**
     * Adds an icon based on the file type "dot" extension
     * string, e.g: ".gif". Case is ignored.
     */
    public void putIcon(String extension, Icon icon) {
        icons.put(extension, icon);
    }

    /**
     * Icon that reperesents this file. Default implementation returns
     * null. You might want to override this to return something more
     * interesting.
     *
     * @see FileView#getIcon
     */
    public Icon getIcon(File f) {
        Icon icon = null;
        String extension = getExtension(f);
        if(extension != null) {
            icon = (Icon) icons.get(extension);
        }
        return icon;
    }

    /**
     * Whether the file is hidden or not. This implementation returns
     * true if the filename starts with a "."
     *
     * @see FileView#isHidden
     */
    public Boolean isHidden(File f) {
        String name = f.getName();
        if(name != null && !name.equals("") && name.charAt(0) == '.') {
            return Boolean.TRUE;
        }
      else {
            return Boolean.FALSE;
        }
    };

    /**
     * Whether the directory is traversable or not. Generic implementation
     * returns true for all directories.
     *
     * You might want to subtype ExampleFileView to do somethimg more
     * interesting, such as recognize compound documents directories; in such a
     * case you might return a special icon for the diretory that makes it look
     * like a regular document, and return false for isTraversable to not allow
     * users to descend into the directory.
     *
     * @see FileView#isTraversable
     */
    public Boolean isTraversable(File f) {
        if(f.isDirectory()) {
            return Boolean.TRUE;
        }
      else {
            return Boolean.FALSE;
        }
    };

}//end ConvenientFileView class

//eof ConvenientFileView.java
```

4. IDEOptions.java

```
package mil.navy.nps.cs.babel;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

import java.io.*;
import org.exolab.castor.xml.ClassDescriptorResolver;
import org.exolab.castor.xml.Unmarshaller;
import org.exolab.castor.xml.Marshaller;
import org.exolab.castor.xml.MarshalException;
import org.exolab.castor.xml.util.ClassDescriptorResolverImpl;


/**
 * Title:        Babel
 * Description:  Controls storage/retreval/updating of options for IDE
 * Copyright:    Copyright (c) 2001
 * Company:      Naval Postgraduate School
 * @author Major Brent P. Christie USMC
 * @version 1.0
 */

public class IDEOptions {

   // ***** DATA MEMBERS *****

   private Component parent;

   private JPanel topPanel;

   private JTabbedPane tabbedPane;
   private JPanel pathPanel;
   private JPanel pathPanelLeft;
   private JPanel pathPanelRight;
   private JTextField cssPath;
   private JTextField fiomPath;
   private JButton cssButton;
   private JButton fiomButton;

   private Options options = null;
   private String opFileLocation = "options.xml";
   private String[] actions = { "cssButton", "fiomButton" };

   //private File optionsFile;

   // ***** CONSTUCTORS *****

   public IDEOptions( Component parent ) {

      this.parent = parent;

      try {
         options = Options.unmarshal(new FileReader( opFileLocation ));
      }
      catch (Exception e) {
           e.printStackTrace();
      }

      buildDialog();


      //open stored options file

   }//end IDEOptions() constructor
```

```
// ***** METHODS *****

public void display() {

   String dialogTitle = "IDE Options";
   String dialogButtons[] = { "Update", "Cancel" };

   String lastCSSPath = cssPath.getText();
   String lastFIOMPath = fiomPath.getText();


   int result = JOptionPane.showOptionDialog( parent,
                                          topPanel, dialogTitle,
                                          JOptionPane.OK_CANCEL_OPTION,
                                          JOptionPane.PLAIN_MESSAGE,
                                          null,
                                          dialogButtons,
                                          dialogButtons[0]);

   if ( result == JOptionPane.OK_OPTION && options != null ) {

      options.getPaths().setComponentSystemSchemaPath( cssPath.getText() );
      options.getPaths().setFiomPath( fiomPath.getText() );

      try {
         options.marshal( new FileWriter( opFileLocation ) );
      }
      catch (Exception e) {
            e.printStackTrace();
      }

   }
   else {

      cssPath.setText( lastCSSPath );
      fiomPath.setText( lastFIOMPath );
   }

}//end display method


public String getCSSPath() {
   return cssPath.getText();
}//end getCSSPath method

public String getFIOMPath() {
   return fiomPath.getText();
}//end getFIOMPath()


private void buildDialog() {

   topPanel = new JPanel();
   tabbedPane = new JTabbedPane();
   pathPanel = new JPanel();
   pathPanelLeft = new JPanel();
   pathPanelRight = new JPanel();

   cssPath = new JTextField(15);
   fiomPath = new JTextField(15);


   if ( options != null ) {
      cssPath.setText( options.getPaths().getComponentSystemSchemaPath() );
      fiomPath.setText( options.getPaths().getFiomPath() );
   }

   cssButton = new JButton( "..." );
   fiomButton = new JButton( "..." );
```

```
      cssButton.addActionListener( new CSSPathHandler() );
      fiomButton.addActionListener( new FIOMPathHandler() );


      pathPanelLeft.setLayout(new GridLayout(0, 2));
      pathPanelRight.setLayout(new GridLayout(0, 1));

      pathPanelLeft.add(new JLabel ("Component System Schema(s) Location: "));
      pathPanelLeft.add( cssPath );
      pathPanelRight.add( cssButton );

      pathPanelLeft.add(new JLabel("FIOM Location: "));
      pathPanelLeft.add( fiomPath );
      pathPanelRight.add( fiomButton );

      pathPanel.add( pathPanelLeft );
      pathPanel.add( pathPanelRight );
      tabbedPane.add( "Paths", pathPanel );
      topPanel.add( tabbedPane );

   }//end buildDialog method

   private class CSSPathHandler implements ActionListener {
      public void actionPerformed( ActionEvent e ) {

         JFileChooser jfc = new JFileChooser();
         jfc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);

         if ( options != null ) {
            jfc.setCurrentDirectory( new File(
                          options.getPaths().getComponentSystemSchemaPath() ) );
         }


         int result = jfc.showDialog( parent, "Update Path" );
         if( result == JFileChooser.APPROVE_OPTION ) {

            cssPath.setText( jfc.getSelectedFile().getPath() );

         }

      }//end actionPerformed method
   }//end CSSPathHandler inner class


   private class FIOMPathHandler implements ActionListener {
      public void actionPerformed( ActionEvent e ) {

         JFileChooser jfc = new JFileChooser();
         jfc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);

         if ( options != null ) {
            jfc.setCurrentDirectory( new File(
                                   options.getPaths().getFiomPath() ) );
         }

         int result = jfc.showDialog( parent, "Update Path" );
         if( result == JFileChooser.APPROVE_OPTION ) {

            fiomPath.setText( jfc.getSelectedFile().getPath() );
         }

      }//end actionPerformed method
   }//end FIOMPathHandler inner class



}//end IDEOptions class
//eof IDEOptions.java
```

5. MainFrame.java

```
package mil.navy.nps.cs.babel;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * Title:        Babel
 * Description:  A Integrated Development Enviornment (IDE) for construction of
 *               a Federation Interoperability Object Model (FIOM).  This tool
 *               will enable a Interoperability Engineer (IE) to design and
 *               maintain a federation of real world entities (RWE) based on
 *               component system representations.
 * Copyright:    Copyright (c) 2001
 * Company:      Naval Postgraduate School
 * @author Major Brent P. Christie USMC
 * @version 1.0
 */
public class MainFrame extends JFrame {

   // ***** DATA MEMBERS *****

   JPanel contentPane;

   //menu bar information
   JMenuBar jMenuBar = new JMenuBar();

   JMenu jMenuFile = new JMenu();
   JMenuItem jMenuFileLoad = new JMenuItem();
   JMenuItem jMenuFileClose = new JMenuItem();
   JMenuItem jMenuFileExit = new JMenuItem();

   JMenu jMenuView = new JMenu();
   JCheckBoxMenuItem jMenuViewXMLSchemaTree = new JCheckBoxMenuItem();
   JCheckBoxMenuItem jMenuViewXMLSchemaText = new JCheckBoxMenuItem();

   JMenu jMenuTools = new JMenu();
   JMenuItem jMenuToolsGen = new JMenuItem();
   JMenuItem jMenuToolsIDE = new JMenuItem();

   JMenu jMenuHelp = new JMenu();
   JMenuItem jMenuHelpAbout = new JMenuItem();

   BorderLayout borderLayout1 = new BorderLayout();

   JPanel statusBar = new JPanel();
   JLabel statusGeneralText = new JLabel();
   JLabel statusSchemaText = new JLabel();

   JSplitPane viewSearchSplit = new JSplitPane();
   JSplitPane schemaFiomSplit = new JSplitPane();

   JTabbedPane schemaTabbedPane = new JTabbedPane();

   IDEOptions ideOptions = new IDEOptions( this );
   Schema schema = new Schema( this, ideOptions );

   // ***** CONSTUCTORS *****

   /**
    * Construct the frame
    */
   public MainFrame() {
      enableEvents( AWTEvent.WINDOW_EVENT_MASK );
      try {
```

```
      init();
   }
   catch( Exception e ) {
      e.printStackTrace();
   }
}//end MainFrame constructor


// ***** METHODS *****

/**
 * Component initialization
 */
private void init() throws Exception  {
   this.setIconImage( Toolkit.getDefaultToolkit().createImage
            ( MainFrame.class.getResource( "images/babel.jpg" ) ) );

   this.setSize(new Dimension( 800, 600 ) );
   this.setTitle( "Babel - Federation Management and Component System " +
                  "Integration Tool" );

   // Set up Menu bar
   jMenuFile.setText( "File" );
   jMenuFile.setMnemonic( 'F' );
   jMenuFileExit.setText( "Exit" );
   jMenuFileExit.setMnemonic( 'x' );
   jMenuFileExit.addActionListener( new ActionListener()  {
      public void actionPerformed( ActionEvent e ) {
         jMenuFileExit_actionPerformed( e );
      }
   });


   jMenuFileLoad.setText( "Load XML Schema File" );
   jMenuFileLoad.setMnemonic( 'L' );
   jMenuFileLoad.addActionListener( new ActionListener() {
      public void actionPerformed( ActionEvent e ) {
         jMenuFileLoad_actionPerformed( e );
      }
   });


   jMenuFileClose.setText( "Close XML Schema File" );
   jMenuFileClose.setMnemonic( 'C' );
   jMenuFileClose.setEnabled( false );
   jMenuFileClose.addActionListener( new ActionListener() {
      public void actionPerformed( ActionEvent e ) {
         jMenuFileClose_actionPerformed( e );
      }
   });


   jMenuView.setText( "View" );
   jMenuView.setMnemonic( 'V' );
   jMenuViewXMLSchemaTree.setEnabled( false );
   jMenuViewXMLSchemaTree.setState( false );
   jMenuViewXMLSchemaTree.setText( "XML Schema Tree" );
   jMenuViewXMLSchemaTree.setMnemonic( 'S' );
   jMenuViewXMLSchemaTree.addItemListener( new ItemListener() {
      public void itemStateChanged( ItemEvent e) {
         jMenuViewXMLSchemaTree_itemStateChanged( e );
      }
   });
   jMenuViewXMLSchemaText.setEnabled( false );
   jMenuViewXMLSchemaText.setState( false );
   jMenuViewXMLSchemaText.setText( "XML Schema Text" );
   jMenuViewXMLSchemaText.setMnemonic( 'X' );
   jMenuViewXMLSchemaText.addItemListener( new ItemListener() {
      public void itemStateChanged( ItemEvent e) {
         jMenuViewXMLSchemaText_itemStateChanged( e );
      }
   });


   jMenuTools.setText( "Tools" );
   jMenuTools.setMnemonic( 'T' );
   jMenuToolsGen.setText( "Generate Java Classes" );
   jMenuToolsGen.setMnemonic( 'G' );
   jMenuToolsGen.setEnabled( false );
   jMenuToolsGen.addActionListener( new ActionListener()  {
      public void actionPerformed( ActionEvent e ) {
         jMenuToolsGen_actionPerformed( e );
      }
   });
   jMenuToolsIDE.setText( "IDE Options" );
   jMenuToolsIDE.setMnemonic( 'O' );
   jMenuToolsIDE.addActionListener( new ActionListener()  {
      public void actionPerformed( ActionEvent e ) {
         jMenuToolsIDE_actionPerformed( e );
      }
   });



   jMenuHelp.setText( "Help" );
   jMenuHelp.setMnemonic( 'H' );
   jMenuHelpAbout.setText( "About" );
   jMenuHelpAbout.setMnemonic( 'A' );
   jMenuHelpAbout.addActionListener( new ActionListener()  {
      public void actionPerformed( ActionEvent e ) {
         jMenuHelpAbout_actionPerformed( e );
      }
   });


   jMenuFile.add( jMenuFileLoad );
   jMenuFile.add( jMenuFileClose );
   jMenuFile.add( jMenuFileExit );
   jMenuView.add( jMenuViewXMLSchemaTree );
   jMenuView.add( jMenuViewXMLSchemaText );
   jMenuTools.add( jMenuToolsGen );
   jMenuTools.add( jMenuToolsIDE );
   jMenuHelp.add( jMenuHelpAbout );
   jMenuBar.add( jMenuFile );
   jMenuBar.add( jMenuView );
   jMenuBar.add( jMenuTools );
   jMenuBar.add( jMenuHelp );

   this.setJMenuBar( jMenuBar );

   // set up contentPane
   contentPane = ( JPanel ) this.getContentPane();
   contentPane.setLayout( borderLayout1 );
   contentPane.setBackground( Color.white );


   // set up status bar
   statusGeneralText.setText( "Let's Babel!  ");
   statusBar.add( statusGeneralText );
   contentPane.add( statusBar, borderLayout1.SOUTH );

   // set up schema / fiom split pane
   schemaFiomSplit.setOrientation( JSplitPane.HORIZONTAL_SPLIT );
   schemaFiomSplit.setLeftComponent( schemaTabbedPane );
   schemaFiomSplit.setRightComponent( null );


   // set up view / search split pane
   viewSearchSplit.setOrientation( JSplitPane.VERTICAL_SPLIT );
   viewSearchSplit.setTopComponent( schemaFiomSplit );
   viewSearchSplit.setBottomComponent( null );
   contentPane.add( viewSearchSplit, borderLayout1.CENTER );
```

```
   show();

}//end init method


/**
 * Loads a new XML schema from a .xsd file
 */
void jMenuFileLoad_actionPerformed( ActionEvent e ) {
   if ( schema.loadFile() ) {
      statusSchemaText.setText( "Schema path -->" +
                                schema.getFile().getPath() );
      statusBar.add( statusSchemaText );
      jMenuFileLoad.setEnabled( false );
      jMenuFileClose.setEnabled( true );
      jMenuViewXMLSchemaTree.setEnabled( true );
      jMenuViewXMLSchemaText.setEnabled( true );
      jMenuToolsGen.setEnabled( true );
      show();
   }
}///end jMenuFileLoad_actionPerformed method


/**
 * Closes XML schema file and all related windows
 */
void jMenuFileClose_actionPerformed( ActionEvent e ) {
   schema.closeFile();
   statusBar.remove( statusSchemaText );
   jMenuFileLoad.setEnabled( true );
   jMenuFileClose.setEnabled( false );
   jMenuViewXMLSchemaTree.setState( false );
   jMenuViewXMLSchemaTree.setEnabled( false );
   jMenuViewXMLSchemaText.setState( false );
   jMenuViewXMLSchemaText.setEnabled( false );
   jMenuToolsGen.setEnabled( false );
   show();
}///end jMenuFileLoad_actionPerformed method


/**
 * File | Exit action performed
 */
public void jMenuFileExit_actionPerformed( ActionEvent e ) {
   System.exit( 0 );
}//end jMenuFileExit_actionPerformed method


/**
 * Displays view of loaded shema file
 */
void jMenuViewXMLSchemaTree_itemStateChanged( ItemEvent e ) {
   if ( e.getStateChange() == ItemEvent.SELECTED ) {
      schemaTabbedPane.addTab( "Schema Tree View",
                             schema.getTreeView( true ) );
      show();
   }
   else {
      schemaTabbedPane.remove( schema.getTreeView( true ) );
   }
}///end jMenuViewXMLSchemaTree_actionPerformed method


/**
 * Displays view of loaded shema file
 */
void jMenuViewXMLSchemaText_itemStateChanged( ItemEvent e ) {

   if ( e.getStateChange() == ItemEvent.SELECTED ) {
      schemaTabbedPane.addTab( "Schema Text View",
                                 schema.getTextView( false ) );
            show();
         }
         else {
            schemaTabbedPane.remove( schema.getTextView( false ) );
         }
      }///end jMenuViewXMLSchemaTree_actionPerformed method


      /**
       * Generates Java classes from open XML Schema file
       */
      void jMenuToolsGen_actionPerformed( ActionEvent e ) {
         schema.generateJavaClasses();
      }///end jMenuToolsIDE_actionPerformed method


      /**
       *  Displays IDE Options
       */
      void jMenuToolsIDE_actionPerformed( ActionEvent e ) {
         ideOptions.display();
      }///end jMenuToolsIDE_actionPerformed method


      /**
       * Help | About action performed
       */
      public void jMenuHelpAbout_actionPerformed( ActionEvent e ) {
         MainFrame_AboutBox dlg = new MainFrame_AboutBox( this );
      }//end jMenuHelpAbout_actionPerformed method


      /**
       * Overridden so we can exit when window is closed
       */
      protected void processWindowEvent( WindowEvent e ) {
         super.processWindowEvent( e );
         if ( e.getID() == WindowEvent.WINDOW_CLOSING ) {
            jMenuFileExit_actionPerformed( null );
         }
      }//end processWindowEvent method



   }//end MainFrame class

   //eof MainFrame.java
```

6. MainFrame_AboutBox.java

```
package mil.navy.nps.cs.babel;

import java.awt.*;
import javax.swing.*;

/**
 * Title:        Babel
 * Description:  Displays about information
 * Copyright:    Copyright (c) 2001
 * Company:      Naval Postgraduate School
 * @author Major Brent P. Christie USMC
 * @version 1.0
 */

public class MainFrame_AboutBox extends JOptionPane {

   // ***** DATA MEMBERS *****
   private String msg = "Babel \n" +
                "Version 1.0 \n" +
                "Copyright (c) 2001 \n\n" +
                "Authors: \n" +
                "     CAPTAIN Paul E. Young USN \n" +
                "     Major Brent P. Christie USMC \n" +
                "     Captain Randy G. Pugh USMC \n\n" +
                "Overview: \n" +
                "      A Integrated Development Enviornment (IDE) for \n" +
                "construction of a Federation Interoperability Object \n" +
                "Model (FIOM).  This tool will enable a Interoperability \n" +
                "Engineer (IE) to design and maintain a federation of \n" +
                "real world entities (RWE) based on component system \n" +
                "representations. All based on Young's OOMI!";

   private ImageIcon aboutIcon = new ImageIcon
                ( MainFrame_AboutBox.class.getResource( "images/babel.jpg" ) );

   // ***** CONSTUCTORS *****

   public MainFrame_AboutBox( Component parent ) {
      super();

      showMessageDialog( parent,
                         msg,
                         "About",
                         PLAIN_MESSAGE,
                         aboutIcon );

   }//end MainFrame_AboutBox constructor


   // ***** METHODS *****

      // none.

}//end MainFrame_AboutBox class

//eof MainFrame_AboutBox.java
```

7. Options.java

```
/*
 * This class was automatically generated with
 * <a href="http://castor.exolab.org">Castor 0.9.2</a>, using an
 * XML Schema.
 * $Id: Options.java,v 1.1.1.1 2001/08/27 23:42:30 bpchrist Exp $
 */

package mil.navy.nps.cs.babel;

  //---------------------------------/
 //- Imported classes and packages -/
//---------------------------------/

import java.io.Reader;
import java.io.Serializable;
import java.io.Writer;
import org.exolab.castor.xml.*;
import org.exolab.castor.xml.MarshalException;
import org.exolab.castor.xml.ValidationException;
import org.xml.sax.DocumentHandler;

/**
 *
 * @version $Revision: 1.1.1.1 $ $Date: 2001/08/27 23:42:30 $
**/
public class Options implements java.io.Serializable {
```

```
              //--------------------------/
             //- Class/Member Variables -/
            //--------------------------/

            private Paths _paths;


              //----------------/
             //- Constructors -/
            //----------------/

            public Options() {
                super();
            } //-- mil.navy.nps.cs.babel.Options()


              //-----------/
             //- Methods -/
            //-----------/

            /**
            **/
            public Paths getPaths()
            {
                return this._paths;
            } //-- Paths getPaths()

            /**
            **/
            public boolean isValid()
            {
                try {
                    validate();
                }
                catch (org.exolab.castor.xml.ValidationException vex) {
                    return false;
                }
                return true;
            } //-- boolean isValid()

            /**
             *
             * @param out
            **/
            public void marshal(java.io.Writer out)
                throws org.exolab.castor.xml.MarshalException,
org.exolab.castor.xml.ValidationException
            {

                Marshaller.marshal(this, out);
            } //-- void marshal(java.io.Writer)

            /**
             *
             * @param handler
            **/
            public void marshal(org.xml.sax.DocumentHandler handler)
                throws org.exolab.castor.xml.MarshalException,
org.exolab.castor.xml.ValidationException
            {

                Marshaller.marshal(this, handler);
            } //-- void marshal(org.xml.sax.DocumentHandler)

            /**
             *
             * @param paths
            **/
            public void setPaths(Paths paths)
            {
                this._paths = paths;
```

```
    } //-- void setPaths(Paths)

    /**
     *
     * @param reader
    **/
    public static mil.navy.nps.cs.babel.Options unmarshal(java.io.Reader reader)
        throws org.exolab.castor.xml.MarshalException,
org.exolab.castor.xml.ValidationException
    {
        return (mil.navy.nps.cs.babel.Options)
Unmarshaller.unmarshal(mil.navy.nps.cs.babel.Options.class, reader);
    } //-- mil.navy.nps.cs.babel.Options unmarshal(java.io.Reader)

    /**
    **/
    public void validate()
        throws org.exolab.castor.xml.ValidationException
    {
        org.exolab.castor.xml.Validator validator = new
org.exolab.castor.xml.Validator();
        validator.validate(this);
    } //-- void validate()

}
```

8. OptionsDescriptor.java

```
/*
 * This class was automatically generated with
 * <a href="http://castor.exolab.org">Castor 0.9.2</a>, using an
 * XML Schema.
 * $Id: OptionsDescriptor.java,v 1.1.1.1 2001/08/27 23:42:30 bpchrist Exp $
 */

package mil.navy.nps.cs.babel;

  //---------------------------------/
 //- Imported classes and packages -/
//---------------------------------/

import org.exolab.castor.mapping.AccessMode;
import org.exolab.castor.mapping.ClassDescriptor;
import org.exolab.castor.mapping.FieldDescriptor;
import org.exolab.castor.xml.*;
import org.exolab.castor.xml.FieldValidator;
import org.exolab.castor.xml.TypeValidator;
import org.exolab.castor.xml.XMLFieldDescriptor;
import org.exolab.castor.xml.handlers.*;
import org.exolab.castor.xml.util.XMLFieldDescriptorImpl;
import org.exolab.castor.xml.validators.*;

/**
 *
 * @version $Revision: 1.1.1.1 $ $Date: 2001/08/27 23:42:30 $
**/
public class OptionsDescriptor extends
org.exolab.castor.xml.util.XMLClassDescriptorImpl {


      //--------------------------/
     //- Class/Member Variables -/
    //--------------------------/

    private java.lang.String nsPrefix;

    private java.lang.String nsURI;

    private java.lang.String xmlName;
    private org.exolab.castor.xml.XMLFieldDescriptor identity;


      //----------------/
     //- Constructors -/
    //----------------/

    public OptionsDescriptor() {
        super();
        xmlName = "options";
        XMLFieldDescriptorImpl  desc           = null;
        XMLFieldHandler         handler        = null;
        FieldValidator          fieldValidator = null;
        //-- initialize attribute descriptors

        //-- initialize element descriptors

        //-- _paths
        desc = new XMLFieldDescriptorImpl(Paths.class, "_paths", "paths",
    NodeType.Element);
        handler = (new XMLFieldHandler() {
            public Object getValue( Object object )
                throws IllegalStateException
            {
                Options target = (Options) object;
                return target.getPaths();
            }
            public void setValue( Object object, Object value)
                throws IllegalStateException, IllegalArgumentException
            {
                try {
                    Options target = (Options) object;
                    target.setPaths( (Paths) value);
                }
                catch (Exception ex) {
                    throw new IllegalStateException(ex.toString());
                }
            }
            public Object newInstance( Object parent ) {
                return new Paths();
            }
        } );
        desc.setHandler(handler);
        desc.setRequired(true);
        desc.setMultivalued(false);
        addFieldDescriptor(desc);

        //-- validation code for: _paths
        fieldValidator = new FieldValidator();
        fieldValidator.setMinOccurs(1);
        desc.setValidator(fieldValidator);

    } //-- mil.navy.nps.cs.babel.OptionsDescriptor()


      //-----------/
     //- Methods -/
    //-----------/

    /**
    **/
    public org.exolab.castor.mapping.AccessMode getAccessMode()
    {
        return null;
    } //-- org.exolab.castor.mapping.AccessMode getAccessMode()

    /**
    **/
    public org.exolab.castor.mapping.ClassDescriptor getExtends()
    {
        return null;
    } //-- org.exolab.castor.mapping.ClassDescriptor getExtends()

    /**
    **/
    public org.exolab.castor.mapping.FieldDescriptor getIdentity()
    {
        return identity;
    } //-- org.exolab.castor.mapping.FieldDescriptor getIdentity()

    /**
    **/
    public java.lang.Class getJavaClass()
    {
        return mil.navy.nps.cs.babel.Options.class;
    } //-- java.lang.Class getJavaClass()

    /**
    **/
    public java.lang.String getNameSpacePrefix()
    {
        return nsPrefix;
    } //-- java.lang.String getNameSpacePrefix()

    /**
    **/
    public java.lang.String getNameSpaceURI()
    {
        return nsURI;
    } //-- java.lang.String getNameSpaceURI()

    /**
    **/
    public org.exolab.castor.xml.TypeValidator getValidator()
    {
        return this;
    } //-- org.exolab.castor.xml.TypeValidator getValidator()

    /**
    **/
    public java.lang.String getXMLName()
    {
        return xmlName;
    } //-- java.lang.String getXMLName()

}
```

9. Paths.java

```
/*
 * This class was automatically generated with
 * <a href="http://castor.exolab.org">Castor 0.9.2</a>, using an
 * XML Schema.
 * $Id: Paths.java,v 1.1.1.1 2001/08/27 23:42:30 bpchrist Exp $
 */

package mil.navy.nps.cs.babel;

  //---------------------------------/
 //- Imported classes and packages -/
//---------------------------------/

import java.io.Reader;
import java.io.Serializable;
import java.io.Writer;
import org.exolab.castor.xml.*;
import org.exolab.castor.xml.MarshalException;
import org.exolab.castor.xml.ValidationException;
import org.xml.sax.DocumentHandler;

/**
```

```
 *
 * @version $Revision: 1.1.1.1 $ $Date: 2001/08/27 23:42:30 $
**/
public class Paths implements java.io.Serializable {


      //--------------------------/
     //- Class/Member Variables -/
    //--------------------------/

    private java.lang.String _componentSystemSchemaPath;

    private java.lang.String _fiomPath;


      //----------------/
     //- Constructors -/
    //----------------/

    public Paths() {
        super();
    } //-- mil.navy.nps.cs.babel.Paths()


      //-----------/
     //- Methods -/
    //-----------/

    /**
    **/
    public java.lang.String getComponentSystemSchemaPath()
    {
        return this._componentSystemSchemaPath;
    } //-- java.lang.String getComponentSystemSchemaPath()

    /**
    **/
    public java.lang.String getFiomPath()
    {
        return this._fiomPath;
    } //-- java.lang.String getFiomPath()

    /**
    **/
    public boolean isValid()
    {
        try {
            validate();
        }
        catch (org.exolab.castor.xml.ValidationException vex) {
            return false;
        }
        return true;
    } //-- boolean isValid()

    /**
     *
     * @param out
    **/
    public void marshal(java.io.Writer out)
        throws org.exolab.castor.xml.MarshalException,
org.exolab.castor.xml.ValidationException
    {

        Marshaller.marshal(this, out);
    } //-- void marshal(java.io.Writer)

    /**
     *
     * @param handler
    **/
    public void marshal(org.xml.sax.DocumentHandler handler)
        throws org.exolab.castor.xml.MarshalException,
org.exolab.castor.xml.ValidationException
    {

        Marshaller.marshal(this, handler);
    } //-- void marshal(org.xml.sax.DocumentHandler)

    /**
     *
     * @param componentSystemSchemaPath
    **/
    public void setComponentSystemSchemaPath(java.lang.String
componentSystemSchemaPath)
    {
        this._componentSystemSchemaPath = componentSystemSchemaPath;
    } //-- void setComponentSystemSchemaPath(java.lang.String)

    /**
     *
     * @param fiomPath
    **/
    public void setFiomPath(java.lang.String fiomPath)
    {
        this._fiomPath = fiomPath;
    } //-- void setFiomPath(java.lang.String)

    /**
     *
     * @param reader
    **/
    public static mil.navy.nps.cs.babel.Paths unmarshal(java.io.Reader reader)
        throws org.exolab.castor.xml.MarshalException,
org.exolab.castor.xml.ValidationException
    {
        return (mil.navy.nps.cs.babel.Paths)
Unmarshaller.unmarshal(mil.navy.nps.cs.babel.Paths.class, reader);
    } //-- mil.navy.nps.cs.babel.Paths unmarshal(java.io.Reader)

    /**
    **/
    public void validate()
        throws org.exolab.castor.xml.ValidationException
    {
        org.exolab.castor.xml.Validator validator = new
org.exolab.castor.xml.Validator();
        validator.validate(this);
    } //-- void validate()

}
```

10. PathsDescriptor.java

```
/*
 * This class was automatically generated with
 * <a href="http://castor.exolab.org">Castor 0.9.2</a>, using an
 * XML Schema.
 * $Id: PathsDescriptor.java,v 1.1.1.1 2001/08/27 23:42:30 bpchrist Exp $
 */

package mil.navy.nps.cs.babel;

  //---------------------------------/
 //- Imported classes and packages -/
//---------------------------------/

import org.exolab.castor.mapping.AccessMode;
import org.exolab.castor.mapping.ClassDescriptor;
import org.exolab.castor.mapping.FieldDescriptor;
```

```
        import org.exolab.castor.xml.*;
        import org.exolab.castor.xml.FieldValidator;
        import org.exolab.castor.xml.TypeValidator;
        import org.exolab.castor.xml.XMLFieldDescriptor;
        import org.exolab.castor.xml.handlers.*;
        import org.exolab.castor.xml.util.XMLFieldDescriptorImpl;
        import org.exolab.castor.xml.validators.*;

        /**
         *
         * @version $Revision: 1.1.1.1 $ $Date: 2001/08/27 23:42:30 $
        **/
        public class PathsDescriptor extends
org.exolab.castor.xml.util.XMLClassDescriptorImpl {


              //--------------------------/
             //- Class/Member Variables -/
            //--------------------------/

            private java.lang.String nsPrefix;

            private java.lang.String nsURI;

            private java.lang.String xmlName;

            private org.exolab.castor.xml.XMLFieldDescriptor identity;


              //----------------/
             //- Constructors -/
            //----------------/

            public PathsDescriptor() {
                super();
                xmlName = "paths";
                XMLFieldDescriptorImpl  desc           = null;
                XMLFieldHandler         handler        = null;
                FieldValidator          fieldValidator = null;
                //-- initialize attribute descriptors

                //-- initialize element descriptors

                //-- _componentSystemSchemaPath
                desc = new XMLFieldDescriptorImpl(java.lang.String.class,
"_componentSystemSchemaPath", "componentSystemSchemaPath", NodeType.Element);
                desc.setImmutable(true);
                handler = (new XMLFieldHandler() {
                    public Object getValue( Object object )
                        throws IllegalStateException
                    {
                        Paths target = (Paths) object;
                        return target.getComponentSystemSchemaPath();
                    }
                    public void setValue( Object object, Object value)
                        throws IllegalStateException, IllegalArgumentException
                    {
                        try {
                            Paths target = (Paths) object;
                            target.setComponentSystemSchemaPath( (java.lang.String)
value);
                        }
                        catch (Exception ex) {
                            throw new IllegalStateException(ex.toString());
                        }
                    }
                    public Object newInstance( Object parent ) {
                        return null;
                    }
                } );
                desc.setHandler(handler);
                desc.setRequired(true);
                desc.setMultivalued(false);
                addFieldDescriptor(desc);

                //-- validation code for: _componentSystemSchemaPath
                fieldValidator = new FieldValidator();
                fieldValidator.setMinOccurs(1);
                { //-- local scope
                    StringValidator sv = new StringValidator();
                    sv.setWhiteSpace("preserve");
                    fieldValidator.setValidator(sv);
                }
                desc.setValidator(fieldValidator);

                //-- _fiomPath
                desc = new XMLFieldDescriptorImpl(java.lang.String.class, "_fiomPath",
"fiomPath", NodeType.Element);
                desc.setImmutable(true);
                handler = (new XMLFieldHandler() {
                    public Object getValue( Object object )
                        throws IllegalStateException
                    {
                        Paths target = (Paths) object;
                        return target.getFiomPath();
                    }
                    public void setValue( Object object, Object value)
                        throws IllegalStateException, IllegalArgumentException
                    {
                        try {
                            Paths target = (Paths) object;
                            target.setFiomPath( (java.lang.String) value);
                        }
                        catch (Exception ex) {
                            throw new IllegalStateException(ex.toString());
                        }
                    }
                    public Object newInstance( Object parent ) {
                        return null;
                    }
                } );
                desc.setHandler(handler);
                desc.setRequired(true);
                desc.setMultivalued(false);
                addFieldDescriptor(desc);

                //-- validation code for: _fiomPath
                fieldValidator = new FieldValidator();
                fieldValidator.setMinOccurs(1);
                { //-- local scope
                    StringValidator sv = new StringValidator();
                    sv.setWhiteSpace("preserve");
                    fieldValidator.setValidator(sv);
                }
                desc.setValidator(fieldValidator);

            } //-- mil.navy.nps.cs.babel.PathsDescriptor()


              //-----------/
             //- Methods -/
            //-----------/

            /**
            **/
            public org.exolab.castor.mapping.AccessMode getAccessMode()
            {
                return null;
            } //-- org.exolab.castor.mapping.AccessMode getAccessMode()

            /**
            **/
```

```
    public org.exolab.castor.mapping.ClassDescriptor getExtends()
    {
        return null;
    } //-- org.exolab.castor.mapping.ClassDescriptor getExtends()

    /**
    **/
    public org.exolab.castor.mapping.FieldDescriptor getIdentity()
    {
        return identity;
    } //-- org.exolab.castor.mapping.FieldDescriptor getIdentity()

    /**
    **/
    public java.lang.Class getJavaClass()
    {
        return mil.navy.nps.cs.babel.Paths.class;
    } //-- java.lang.Class getJavaClass()

    /**
    **/
    public java.lang.String getNameSpacePrefix()
    {
        return nsPrefix;
    } //-- java.lang.String getNameSpacePrefix()

    /**
    **/
    public java.lang.String getNameSpaceURI()
    {
        return nsURI;
    } //-- java.lang.String getNameSpaceURI()

    /**
    **/
    public org.exolab.castor.xml.TypeValidator getValidator()
    {
        return this;
    } //-- org.exolab.castor.xml.TypeValidator getValidator()

    /**
    **/
    public java.lang.String getXMLName()
    {
        return xmlName;
    } //-- java.lang.String getXMLName()

}
```

11. Schema.java

```
package mil.navy.nps.cs.babel;

import java.awt.*;
import java.io.*;
import java.net.URL;
import javax.swing.*;
import javax.swing.filechooser.*;
import javax.swing.text.html.*;
import xbrowser.renderer.custom.*;

import org.exolab.castor.builder.*;

import javax.xml.parsers.*;

import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

import org.w3c.dom.Document;
import org.w3c.dom.DOMException;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerConfigurationException;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

/**
 * Title:        Babel
 * Description:  Handles all schema file operations
 * Copyright:    Copyright (c) 2001
 * Company:      Naval Postgraduate School
 * @author Major Brent P. Christie USMC
 * @version 1.0
 *
 */
public class Schema {

   // ***** DATA MEMBERS *****

   private Component parent;
   private IDEOptions options;


   private ConvenientFileFilter filter;
   private String schemaFilter = "xsd";
   private String schemaDescription = "XML Schema Files";

   private ConvenientFileView fileView;

   private File schemaFile = null;
   private boolean schemaFileLoaded = false;

   private Document document, documentWithInclude;

   private JScrollPane treeView, treeViewInc,
                       textView, textViewInc;


   // ***** CONSTUCTORS *****

   /**
    * Schema constructor
    * @param parent parent window
    * @param options contains file path information
    */
   public Schema(Component parent, IDEOptions options) {

      this.parent = parent;
      this.options = options;

   }//end Schema constructor


   // ***** METHODS *****

   /**
    * Loads schema file using JFileChooser.
    * @return True if file loaded properly
    */
   public boolean loadFile() {

      JFileChooser chooser = new JFileChooser( options.getCSSPath() );
      chooser.setFileSelectionMode(JFileChooser.FILES_ONLY);

      //set file filter(s)
      filter = new ConvenientFileFilter( schemaFilter, schemaDescription );
      chooser.addChoosableFileFilter( filter );

      //set icons(s)
      fileView = new ConvenientFileView();
      fileView.putIcon( schemaFilter,
                        new ImageIcon( Schema.class.getResource
                                              ( "images/babel_Icon.jpg" ) ) );
      chooser.setFileView( fileView );


      while ( true ) {

         int result = chooser.showDialog( parent, "Load" );


         if ( result == JFileChooser.APPROVE_OPTION ) {

            schemaFile = chooser.getSelectedFile();
            SchemaDocument sd = new SchemaDocument();
            if ( sd.goodLoadFromFile( schemaFile, true ) ) {
               documentWithInclude = sd.getDocument();

               sd.goodLoadFromFile( schemaFile, false );
               document = sd.getDocument();

               treeView = generateTreeView( document );
               treeViewInc = generateTreeView( documentWithInclude );

               textView = generateTextView( document );
               textViewInc = generateTextView( documentWithInclude );

               schemaFileLoaded = true;
               return true;
            }
            else {
               JOptionPane.showMessageDialog( parent,
               "File not valid XML document.  See console messages for " +
                                                    "details.  Try again. ");
            }

         }
         else if ( result == JFileChooser.ERROR_OPTION ) {

            JOptionPane.showMessageDialog( parent, "An file error occured.  " +
                                                 "Try again. ");
         }
         else if ( result == JFileChooser.CANCEL_OPTION ) {
            return false;
         }
         else {
            JOptionPane.showMessageDialog( parent, "Unknown operation " +
                                                 "occured.  Try again. ");
         }

      }//end while
   }//end loadFile method


   /**
    * Frees current file
    */
   public void closeFile() {
      schemaFile = null;
   }//end closeFile method


   /**
    * Returns Loaded XML Schema file chosen by the user
    * @return Loaded XML Schema file chosen
    */
```

```
public File getFile() {
   return schemaFile;
}//end getFile method


/**
 * Uses Castor Source Generator data binding API to create light weight java
 * classes based on input XML schema file.
 * @return True if light wieght classes were generated, False if error
 */
public boolean generateJavaClasses() {

   if ( schemaFileLoaded && schemaFile != null ) {

      String packageName = "default";  //incase user cancels input
      String input = JOptionPane.showInputDialog
                      ( parent, "Enter package name for generated classes" );

      if ( input != null ) {
         packageName = input;
      }

      SourceGenerator sg = new SourceGenerator();
      try {
         sg.generateSource( schemaFile.getPath(), packageName);
         return true;

      }
      catch (java.io.FileNotFoundException e) {
         System.out.println("File not found: " + e);
         return false;
      }

   }
   else {
      return false;
   }

}//end generateJavaClasses method


/**
 * Returns a JScrollPane containing a JTree view of the loaded XML Schema
 * File.
 * @param include If the tree returned is to contain information contained
 * within the files in <include> elements.
 * @return The tree view of the loaded XML Schema file.
 */
public JScrollPane getTreeView( boolean include ) {

   return include ? treeViewInc : textView;

}//end getTreeView method


/**
 * Returns a JScrollPane containing a JEditorPane with a XHTML view of the
 * loaded XML Schema File.
 * @param include True, if the XHTML returned is to contain schema
 * information from the files in <include> elements.
 * @return The tree view of the loaded XML Schema file.
 */
public JScrollPane getTextView( boolean include ) {

   return include ? textViewInc : textView;

}//end getTextView method


/**
 * Returns a JScrollPane containing a JTree view of the document given.
 * @param document document used to create JTree
 * @return Returns a JScrollPane containing a JTree view of the document
 * given.
 */
private JScrollPane generateTreeView( Document document ) {


   // This JScrollPane is the container for the JTree
   JScrollPane          jScrollPane = new JScrollPane();
   // This is the XTree object which displays the XML in a JTree
   XTree                xTree;

   // Build the XTree object
   xTree = new XTree( document );

   // Wrap the XTree in a JScroll so that we can scroll it in the JFrame.
   jScrollPane.getViewport().add( xTree );

   return jScrollPane;

}//end getTreeView method


/**
 * Returns JScrollPane containing a JEditorPane with the XHTML version of the
 * document given.
 * @param document document to convert to XHTML
 * @return Returns JScrollPane containing a JEditorPane with the XHTML
 * version of the document given.
 */
private JScrollPane generateTextView( Document document ) {


   JScrollPane jScrollPane = new JScrollPane();

   XHTMLEditorKit htmlKit = new XHTMLEditorKit();
   JEditorPane jep = new JEditorPane();
   htmlKit.install( jep );
   jep.setEditorKit( htmlKit );

   String xsltStylesheet =  "xmlverbatim.xsl";

   if ( schemaFile != null ) {

      TransformerFactory tFactory = TransformerFactory.newInstance();

      try {

         // Use a Transformer for output

         Transformer transformer = tFactory.newTransformer(
                                        new StreamSource( xsltStylesheet ) );

         DOMSource source = new DOMSource( document );

         // create a new string writer
         StringWriter sw = new StringWriter();

         StreamResult result = new StreamResult( sw );
         transformer.transform( source, result );

         // set the text of the text pane
         //jTextPane.setText( sw.toString() );
         jep.setText( sw.toString() );
      }
      catch ( Exception e ) {
         e.printStackTrace();
      }

   }//end if
```

```
      jScrollPane.getViewport().add( jep );

      return jScrollPane;

   }////end generateTextView method


}//end Schema class
//eof Schema.java
```

11. SchemaDocument.java

```
package mil.navy.nps.cs.babel;

import java.io.*;
import java.util.Vector;

import javax.xml.parsers.*;

import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

import org.w3c.dom.*;

/**
 * Title:        Babel
 * Description:  Used to create a DOM document from a XML schema and any
 * xsd:include schemas.  Note: This should not be necessary
 * once Xerces J parser starts supporting XML schemas.  Current
 * 2.0 beta does not.
 * Copyright:    Copyright (c) 2001
 * Company:      Naval Postgraduate School
 * @author Major Brent P. Christie USMC
 * @version 1.0
 *
 */
public class SchemaDocument {


   // ***** DATA MEMBERS *****

   private DocumentBuilder builder;
   private DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
   private Document document;

   private boolean includeReference;
   private boolean goodLoad = false;


   // ***** CONSTUCTORS *****

   /**
    * No argument constructor
    */
   public SchemaDocument() {

      factory.setIgnoringElementContentWhitespace( true );

   }//end SchemaDocument constructor


   // ***** METHODS *****

   /**
    * Creates new document from schema '.xsd' file.
    * @param schemaFile XML Schema file to load from
    * @param include True to include files referenced in xsd:include
    * elements' schemaLocation attribute.
```

```
           * @return True if document created properly
           */
          public boolean goodLoadFromFile( File schemaFile, boolean include ) {

             document = null;

             includeReference = include;

             goodLoad = true;

             document = parseDocument( schemaFile );

             return goodLoad;


          }//end goodLoadFromFile method



          /**
           * Returns a DOM Document from a schema .xsd file.  The document will contain
included
           * references if the includeReference flag is set.
           * @param schemaFile .xsd file to input
           * @return Document from given file
           */
          private Document parseDocument( File schemaFile ) {

             Document currentDocument = null;

             try {
                builder = factory.newDocumentBuilder();
                currentDocument = builder.parse( schemaFile );

                if ( includeReference ) {

                   currentDocument = createIncludeDocument( currentDocument,
                                                                         schemaFile );
                }//end if includeReference
             }
             catch ( Exception e ) {
                e.printStackTrace();
                goodLoad = false;
             }//end try-catch

             return currentDocument;

          }//end parseDocument method


          /**
           * Adds given schema file to current document.
           * @param currentDocument The document to add include file infromation to.
           * @param schemaFile The .xsd file to add to the currentDocument
           * @return a Document containing the combination of the given Document and the
given Document file.
           */
          public Document createIncludeDocument( Document currentDocument,
                                                File schemaFile ) {

             Document includeDocument;
             String path;


             //find directory where schema file resides
             path = schemaFile.getPath().substring( 0,
                   schemaFile.getPath().length() - schemaFile.getName().length() );

             //find all include elements
             NodeList includeList =
                                 currentDocument.getElementsByTagName( "xsd:include" );
```

```
   for ( int i = 0; i <  includeList.getLength(); i ++ ) {

      Element includeElement = (Element)includeList.item( 0 );

      String schemaLocation = includeElement.getAttribute( "schemaLocation");

      //if valid schema location
      File includeFile = schemaFile;
      if ( ! schemaLocation.equals( "" ) ) {

         //if not absolute path, determine path, else no change
         includeFile = new File( schemaLocation );
         if ( ! includeFile.isAbsolute() ) {

            //Update schemaLocation
            schemaLocation = path + schemaLocation;
            includeFile = new File( schemaLocation );
            try {
               //add relative path to current path
               path = includeFile.getCanonicalPath().substring( 0,
                      includeFile.getCanonicalPath().length() -
                      includeFile.getName().length() );
            }
            catch (IOException ioe) {
               goodLoad = false;
               ioe.printStackTrace();
            }//end try - catch
            includeFile = new File( path + includeFile.getName() );

         }//end if ( ! tempFile.isAbsolute() )

         System.out.println( "Including File " + includeFile.getPath() );

         //recursively search for other documents
         includeDocument = parseDocument( includeFile );

         //merge includeDocument into currentDocument
         createMergeDocument( currentDocument, includeDocument );

      } // end if valid schema location

      //remove found include node and replace with comment
      //<!-- Include file inc... -->
      Node insrtCmt = currentDocument.createComment( "Include file " +
                                          "incorporated and stmt deleted");
      includeElement.getParentNode().replaceChild(insrtCmt,
                                                         includeElement );

   }//end for all include elements loop

   return currentDocument;

}


/**
 * Returns the current DOM Document.
 * @return The current Document.
 */
public Document getDocument() {

   return document;

}//end getDocument method


/**
 * This method merges a include(or any for that matter) document into another
 * (the currentDocument).  It does this by finding all the "type" attributes
 * of all elements in the current document.  For each "type" attribute
 * a search is made in the include document for all elements with a matching
 * "name" attribute.  If found, that element is added as a child of the
 * current document element with matching type.  This allows one to view
 * the guts of a type at the node that references it.
 * @param currentDocument document to update
 * @param includeDocument document to include into current (not changed)
 */
 private void createMergeDocument( Document currentDocument,
                                                Document includeDocument )  {
   Element currentElement;
   String typeAttribute;

   //get included document's root element
   Element currentRoot = currentDocument.getDocumentElement();
   NodeList allChildrenList = currentRoot.getElementsByTagName( "*" );

   for ( int i = 0; i < allChildrenList.getLength(); i++ ) {
      currentElement = (Element)allChildrenList.item(i);
      typeAttribute = currentElement.getAttribute( "type" );

      //separate namespace from type name
      String nameSpace;
      String type = typeAttribute;
      int colon = typeAttribute.indexOf( ":" ); //-1 returned if not found
      if ( colon > 0 ) {
         nameSpace = typeAttribute.substring( 0, colon - 1);
         type = typeAttribute.substring( colon + 1 );
      }


      //add include element as child if match type attribute
      if ( ! type.equals( "" ) ) {

         //first add all complex types from include document
         NodeList complexChild = includeDocument.getElementsByTagName(
                                                         "xsd:complexType" );

         Element complexElement;
         String complexName;
         for (int j = 0; j < complexChild.getLength(); j++ ) {
            complexElement = (Element)complexChild.item( j );
            complexName = complexElement.getAttribute( "name" );
            //if "type" form currentDocument matches "name" from include
            //document add the include element as child of current element
            if ( type.equals( complexName ) ) {
               //create a import node that is a deep copy of the match
               //this will allow view 'guts' of types
               Node importNode = currentDocument.importNode( complexElement,
                                                             true );
               //make sure no duplicates of type information
               if ( !currentElement.hasChildNodes() ) {
                  currentElement.appendChild( importNode );
               }//end if already has type information

            }//end if match
         }//end for complexChild loop


         NodeList simpleChild = includeDocument.getElementsByTagName(
                                                   "xsd:simpleType" );
         Element simpleElement;
         String simpleName;
         for (int j = 0; j < simpleChild.getLength(); j++ ) {
            simpleElement = (Element)simpleChild.item( j );
            simpleName = simpleElement.getAttribute( "name" );
            //if "type" form currentDocument matches "name" from include
            //document add the include element as child of current element
            if ( type.equals( simpleName ) ) {
               //create a import node that is a deep copy of the match
               //this will allow view 'guts' of types
               Node importNode = currentDocument.importNode( simpleElement,
                                                              true );
                    //make sure no duplicates of type information
                    if ( !currentElement.hasChildNodes() ) {
                       currentElement.appendChild( importNode );
                    }//end if alread has type information

                 }//end if match

              }//end for simpleChild loop

           }//end if found type

        }//end for all children in currentDocument

     }///end mergeDoms method

  }//end SchemaDocument class
  //eof SchemaDocument.java
```

12. XTree.java

```
  package mil.navy.nps.cs.babel;

  // W3C DOM classes
  import org.w3c.dom.*;

  // JAXP's classes for DOM I/O
  import javax.xml.parsers.*;

  // Standard Java classes
  import javax.swing.*;
  import javax.swing.tree.*;
  import javax.swing.event.*;
  import java.awt.*;
  import java.awt.event.*;
  import java.io.*;

  /**
   * Description: The XTree class is an extension of the javax.swing.JTree class.
   * It behaves in every way like a JTree component, the difference is that
additional
   * methods have been provided to facilitate the parsing of an XML document into a
   * DOM object and translating that DOM object into a viewable JTree structure.
   *
   * Copyright (c) March 2001 Kyle Gabhart
   * @author Kyle Gabhart
   * @version 1.0
   */

  public class XTree extends JTree
  {
    /**
     * This member stores the TreeNode object used to create the model for the
JTree.
     * The DefaultMutableTreeNode class is defined in the javax.swing.tree package
     * and provides a default implementation of the MutableTreeNode interface.
     */
    private           DefaultMutableTreeNode      treeNode;

    /**
     * These three members are a part of the JAXP API and are used to parse the XML
     * text into a DOM object (of type Document).
     */
    private           Document                    doc;

    /**
     * This single constructor builds an XTree object using the XML text
     * passed in through the constructor.
     *
```

```
         * @param text A String of XML formatted text
         *
         * @exception ParserConfigurationException  This exception is potentially thrown
if
         * the constructor configures the parser improperly.  It won't.
         */
        public XTree( Document document )
        {
            // Initialize the superclass portion of the object
            super();

            doc = document;

            // Set basic properties for the Tree rendering
            getSelectionModel().setSelectionMode(
TreeSelectionModel.SINGLE_TREE_SELECTION );
            setShowsRootHandles( true );
            setEditable( false ); // A more advanced version of this tool would allow
the Tree to be editable


            // Take the DOM root node and convert it to a Tree model for the JTree
            treeNode = createTreeNode( (Node)doc.getDocumentElement() );
            setModel( new DefaultTreeModel( treeNode ) );
        } //end XTree()

         /**
          * This takes a DOM Node and recurses through the children until each one is
added
          * to a DefaultMutableTreeNode. The JTree then uses this object as a tree
model.
          *
          * @param root org.w3c.Node.Node
          *
          * @return Returns a DefaultMutableTreeNode object based on the root Node
passed in
          */
         private DefaultMutableTreeNode createTreeNode( Node root )
         {
            DefaultMutableTreeNode  treeNode = null;
            String                  type, name, value;
            NamedNodeMap            attribs;
            Node                    attribNode;

            // Get data from root node
            type = getNodeType( root );
            name = root.getNodeName();
            value = root.getNodeValue();

            // Special case for TEXT_NODE or COMMENT NODE
            if ( root.getNodeType() == Node.COMMENT_NODE ||
                 root.getNodeType() == Node.TEXT_NODE ) {
               name = value;
            }
            treeNode = new DefaultMutableTreeNode( name );

            // Display the attributes if there are any
            attribs = root.getAttributes();
            if( attribs != null )
            {
               for( int i = 0; i < attribs.getLength(); i++ )
               {
                  attribNode = attribs.item(i);
                  name = attribNode.getNodeName().trim();
                  value = attribNode.getNodeValue().trim();

                  if ( value != null )
                  {
                     if ( value.length() > 0 )
                     {
                        treeNode.add( new DefaultMutableTreeNode( name + "=\"" + value +
  "\"" ) );
                     } //end if ( value.length() > 0 )
                  } //end if ( value != null )
               } //end for( int i = 0; i < attribs.getLength(); i++ )
            } //end if( attribs != null )

            // Recurse children nodes if any exist
            if( root.hasChildNodes() )
            {
               NodeList             children;
               int                  numChildren;
               Node                 node;
               String               data;

               children = root.getChildNodes();
               // Only recurse if Child Nodes are non-null
               if( children != null )
               {
                  numChildren = children.getLength();

                  for (int i=0; i < numChildren; i++)
                  {
                     node = children.item(i);
                     if( node != null )
                     {
                        // A special case could be made for each Node type.
                        if( node.getNodeType() == Node.ELEMENT_NODE )
                        {
                           treeNode.add( createTreeNode(node) );
                        } //end if( node.getNodeType() == Node.ELEMENT_NODE )

                        data = node.getNodeValue();

                        if( data != null )
                        {
                           data = data.trim();
                           if ( !data.equals("\n") && !data.equals("\r\n") &&
  data.length() > 0 )
                           {
                              treeNode.add(createTreeNode(node));
                           } //end if ( !data.equals("\n") && !data.equals("\r\n") &&
  data.length() > 0 )
                        } //end if( data != null )
                     } //end if( node != null )
                  } //end for (int i=0; i < numChildren; i++)
               } //end if( children != null )
            } //end if( root.hasChildNodes() )
            return treeNode;
         } //end createTreeNode( Node root )

         /**
          * This method returns a string representing the type of node passed in.
          *
          * @param node org.w3c.Node.Node
          *
          * @return Returns a String representing the node type
          */
         private String getNodeType( Node node )
         {
            String type;

            switch( node.getNodeType() )
            {
               case Node.ELEMENT_NODE:
               {
                  type = "Element";
                  break;
               }
               case Node.ATTRIBUTE_NODE:
               {
```

```
            type = "Attribute";
            break;
         }
         case Node.TEXT_NODE:
         {
            type = "Text";
            break;
         }
         case Node.CDATA_SECTION_NODE:
         {
            type = "CData section";
            break;
         }
         case Node.ENTITY_REFERENCE_NODE:
         {
            type = "Entity reference";
            break;
         }
         case Node.ENTITY_NODE:
         {
            type = "Entity";
            break;
         }
         case Node.PROCESSING_INSTRUCTION_NODE:
         {
            type = "Processing instruction";
            break;
         }
         case Node.COMMENT_NODE:
         {
            type = "Comment";
            break;
         }
         case Node.DOCUMENT_NODE:
         {
            type = "Document";
            break;
         }
         case Node.DOCUMENT_TYPE_NODE:
         {
            type = "Document type";
            break;
         }
         case Node.DOCUMENT_FRAGMENT_NODE:
         {
            type = "Document fragment";
            break;
         }
         case Node.NOTATION_NODE:
         {
            type = "Notation";
            break;
         }
         default:
         {
            type = "???";
            break;
         }
      }// end switch( node.getNodeType() )
      return type;
   } //end getNodeType()


} //end class XTree
```

B. REQUIRED FILES

1. options.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="options">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="paths">
               <xsd:complexType>
                  <xsd:sequence>
                     <xsd:element name="componentSystemSchemaPath" type="xsd:string"/>
                     <xsd:element name="fiomPath" type="xsd:string"/>
                  </xsd:sequence>
               </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

2. options.xml

```
<?xml version="1.0"?>
<options>
   <paths>
      <componentSystemSchemaPath>C:\XSD_Test\USMTF</componentSystemSchemaPath>
      <fiomPath>c:\Documents and Settings\OOMI Project\babel\fiom</fiomPath>
   </paths>
</options>
```

3. xmlverbatim.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
   XML to HTML Verbatim Formatter with Syntax Highlighting
   Version 1.0.2
   GPL (c) Oliver Becker, 2000-08-12
   obecker@informatik.hu-berlin.de
-->

<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:verb="http://informatik.hu-berlin.de/xmlverbatim"
                exclude-result-prefixes="verb">

   <xsl:output method="html" omit-xml-declaration="yes" indent="no"/>

   <xsl:template match="/">

         <style type="text/css"><!--
         .xmlverb-default          { color: #333333; background-color: #ffffff;
                                     font-family: monospace; }
.xmlverb-element-name     { color: #990000; }
.xmlverb-element-nsprefix { color: #666600; }
.xmlverb-attr-name        { color: #660000; }
.xmlverb-attr-content     { color: #000099; font-weight: bold; }
.xmlverb-ns-name          { color: #666600; }
.xmlverb-ns-uri           { color: #330099; }
.xmlverb-text             { color: #000000; font-weight: bold; }
.xmlverb-comment          { color: #006600; font-style: italic; }
.xmlverb-pi-name          { color: #006600; font-style: italic; }
.xmlverb-pi-content       { color: #006666; font-style: italic; }
         --></style>
   <xsl:apply-templates select="." mode="xmlverb" />

</xsl:template>

<!-- root -->
<xsl:template match="/" mode="xmlverb">
   <xsl:text>&#xA;</xsl:text>
   <xsl:comment>
      <xsl:text> converted by xmlverbatim.xsl 1.0.2, (c) O. Becker </xsl:text>
   </xsl:comment>
   <xsl:text>&#xA;</xsl:text>
   <div class="xmlverb-default">
      <xsl:apply-templates mode="xmlverb">
         <xsl:with-param name="root" select="true()"/>
      </xsl:apply-templates>
   </div>
   <xsl:text>&#xA;</xsl:text>
</xsl:template>

<!-- wrapper -->
<xsl:template match="verb:wrapper">
   <xsl:apply-templates mode="xmlverb" />
</xsl:template>

<xsl:template match="verb:wrapper" mode="xmlverb">
   <xsl:apply-templates mode="xmlverb" />
</xsl:template>

<!-- element nodes -->
<xsl:template match="*" mode="xmlverb">
   <xsl:param name="root" />
   <xsl:text>&lt;</xsl:text>
   <xsl:variable name="ns-prefix"
                 select="substring-before(name(),':')" />
   <xsl:if test="$ns-prefix != ''">
      <span class="xmlverb-element-nsprefix">
         <xsl:value-of select="$ns-prefix"/>
      </span>
      <xsl:text>:</xsl:text>
   </xsl:if>
   <span class="xmlverb-element-name">
      <xsl:value-of select="local-name()"/>
   </span>
   <xsl:variable name="pns" select="../namespace::*"/>
   <xsl:if test="$pns[name()=''] and not(namespace::*[name()=''])">
      <span class="xmlverb-ns-name">
         <xsl:text> xmlns</xsl:text>
      </span>
      <xsl:text>=&quot;&quot;</xsl:text>
   </xsl:if>
   <xsl:for-each select="namespace::*">
      <xsl:if test="not($pns[name()=name(current()) and
                        .=current()])">
         <xsl:call-template name="xmlverb-ns" />
      </xsl:if>
   </xsl:for-each>
   <xsl:for-each select="@*">
      <xsl:call-template name="xmlverb-attrs" />
   </xsl:for-each>
   <xsl:choose>
      <xsl:when test="count(*)=0 and .=''">
         <xsl:text> /&gt;</xsl:text>
      </xsl:when>
      <xsl:otherwise>
         <xsl:text>&gt;</xsl:text>
         <xsl:apply-templates mode="xmlverb" />
         <xsl:text>&lt;/</xsl:text>
         <xsl:if test="$ns-prefix != ''">
            <span class="xmlverb-element-nsprefix">
               <xsl:value-of select="$ns-prefix"/>
            </span>
            <xsl:text>:</xsl:text>
         </xsl:if>
         <span class="xmlverb-element-name">
            <xsl:value-of select="local-name()"/>
         </span>
         <xsl:text>&gt;</xsl:text>
      </xsl:otherwise>
   </xsl:choose>
   <xsl:if test="$root"><br /><xsl:text>&#xA;</xsl:text></xsl:if>
</xsl:template>

<!-- attribute nodes -->
<xsl:template name="xmlverb-attrs">
   <xsl:text> </xsl:text>
   <span class="xmlverb-attr-name">
      <xsl:value-of select="name()"/>
   </span>
   <xsl:text>=&quot;</xsl:text>
   <span class="xmlverb-attr-content">
      <xsl:call-template name="html-replace-entities">
         <xsl:with-param name="text" select="normalize-space(.)" />
         <xsl:with-param name="attrs" select="true()" />
      </xsl:call-template>
   </span>
   <xsl:text>&quot;</xsl:text>
</xsl:template>

<!-- namespace nodes -->
<xsl:template name="xmlverb-ns">
   <xsl:if test="name()!='xml'">
      <span class="xmlverb-ns-name">
         <xsl:text> xmlns</xsl:text>
         <xsl:if test="name()!=''">
            <xsl:text>:</xsl:text>
         </xsl:if>
         <xsl:value-of select="name()"/>
      </span>
      <xsl:text>=&quot;</xsl:text>
      <span class="xmlverb-ns-uri">
         <xsl:value-of select="."/>
      </span>
      <xsl:text>&quot;</xsl:text>
   </xsl:if>
</xsl:template>

<!-- text nodes -->
<xsl:template match="text()" mode="xmlverb">
   <span class="xmlverb-text">
      <xsl:call-template name="preformatted-output">
         <xsl:with-param name="text">
            <xsl:call-template name="html-replace-entities">
               <xsl:with-param name="text" select="." />
            </xsl:call-template>
         </xsl:with-param>
      </xsl:call-template>
   </span>
</xsl:template>

<!-- comments -->
<xsl:template match="comment()" mode="xmlverb">
   <xsl:param name="root" />
   <xsl:text>&lt;!--</xsl:text>
   <span class="xmlverb-comment">
      <xsl:call-template name="preformatted-output">
         <xsl:with-param name="text" select="." />
      </xsl:call-template>
   </span>
   <xsl:text>--&gt;</xsl:text>
   <xsl:if test="$root"><br /><xsl:text>&#xA;</xsl:text></xsl:if>
</xsl:template>
```

```
<!-- processing instructions -->
<xsl:template match="processing-instruction()" mode="xmlverb">
   <xsl:param name="root" />
   <xsl:text>&lt;?</xsl:text>
   <span class="xmlverb-pi-name">
      <xsl:value-of select="name()"/>
   </span>
   <xsl:if test=".!=''">
      <xsl:text> </xsl:text>
      <span class="xmlverb-pi-content">
         <xsl:value-of select="."/>
      </span>
   </xsl:if>
   <xsl:text>?&gt;</xsl:text>
   <xsl:if test="$root"><br /><xsl:text>&#xA;</xsl:text></xsl:if>
</xsl:template>


<!-- ========================================================== -->
<!--                   Procedures / Functions                   -->
<!-- ========================================================== -->

<!-- generate entities by replacing &, ", < and > in $text -->
<xsl:template name="html-replace-entities">
   <xsl:param name="text" />
   <xsl:param name="attrs" />
   <xsl:variable name="tmp">
      <xsl:call-template name="replace-substring">
         <xsl:with-param name="from" select="'&gt;'" />
         <xsl:with-param name="to" select="'&amp;gt;'" />
         <xsl:with-param name="value">
            <xsl:call-template name="replace-substring">
               <xsl:with-param name="from" select="'&lt;'" />
               <xsl:with-param name="to" select="'&amp;lt;'" />
               <xsl:with-param name="value">
                  <xsl:call-template name="replace-substring">
                     <xsl:with-param name="from"
                                     select="'&amp;'" />
                     <xsl:with-param name="to"
                                     select="'&amp;amp;'" />
                     <xsl:with-param name="value"
                                     select="$text" />
                  </xsl:call-template>
               </xsl:with-param>
            </xsl:call-template>
         </xsl:with-param>
      </xsl:call-template>
   </xsl:variable>
   <xsl:choose>
      <!-- $text is an attribute value -->
      <xsl:when test="$attrs">
         <xsl:call-template name="replace-substring">
            <xsl:with-param name="from" select="'&#xA;'" />
            <xsl:with-param name="to" select="'&amp;#xA;'" />
            <xsl:with-param name="value">
               <xsl:call-template name="replace-substring">
                  <xsl:with-param name="from"
                                  select="'&quot;'" />
                  <xsl:with-param name="to"
                                  select="'&amp;quot;'" />
                  <xsl:with-param name="value" select="$tmp" />
               </xsl:call-template>
            </xsl:with-param>
         </xsl:call-template>
      </xsl:when>
      <xsl:otherwise>
         <xsl:value-of select="$tmp" />
      </xsl:otherwise>
   </xsl:choose>
</xsl:template>
```

```
   <!-- replace in $value substring $from with $to -->
   <xsl:template name="replace-substring">
      <xsl:param name="value" />
      <xsl:param name="from" />
      <xsl:param name="to" />
      <xsl:choose>
         <xsl:when test="contains($value,$from)">
            <xsl:value-of select="substring-before($value,$from)" />
            <xsl:value-of select="$to" />
            <xsl:call-template name="replace-substring">
               <xsl:with-param name="value"
                                select="substring-after($value,$from)" />
               <xsl:with-param name="from" select="$from" />
               <xsl:with-param name="to" select="$to" />
            </xsl:call-template>
         </xsl:when>
         <xsl:otherwise>
            <xsl:value-of select="$value" />
         </xsl:otherwise>
      </xsl:choose>
   </xsl:template>

   <!-- preformatted output: space as &nbsp;, tab as 8 &nbsp;
                             nl as <br> -->
   <xsl:template name="preformatted-output">
      <xsl:param name="text" />
      <xsl:call-template name="output-nl">
         <xsl:with-param name="text">
            <xsl:call-template name="replace-substring">
               <xsl:with-param name="value"
                                select="translate($text,' ','&#xA0;')" />
               <xsl:with-param name="from" select="'&#9;'" />
               <xsl:with-param name="to"

select="'&#xA0;&#xA0;&#xA0;&#xA0;&#xA0;&#xA0;&#xA0;&#xA0;'" />
            </xsl:call-template>
         </xsl:with-param>
      </xsl:call-template>
   </xsl:template>

   <!-- output nl as <br> -->
   <xsl:template name="output-nl">
      <xsl:param name="text" />
      <xsl:choose>
         <xsl:when test="contains($text,'&#xA;')">
            <xsl:value-of select="substring-before($text,'&#xA;')" />
            <br />
            <xsl:text>&#xA;</xsl:text>
            <xsl:call-template name="output-nl">
               <xsl:with-param name="text"
                                select="substring-after($text,'&#xA;')" />
            </xsl:call-template>
         </xsl:when>
         <xsl:otherwise>
            <xsl:value-of select="$text" />
         </xsl:otherwise>
      </xsl:choose>
   </xsl:template>

</xsl:stylesheet>
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[Bre01] The Breeze Factor, LLC, "News", [http://www.breezefactor.com/news13.html]. June 2001.

[Cas01] ExoLab Group, "The Source Code Generator", [http://castor.exolab.org/sourcegen.html]. September 2001.

[DII98] U.S. Defense Information Systems Agency, *Defense Information Infrastructure Master Plan*, Version 7.0, Appendix A, Communications and Computer Infrastructure, [http://www.disa.mil/diimp/diimp-a.html]. 13 March 1998.

[DII01] "DII COE DataEmporium." [http://diides.ncr.disa.mil/xmlreg/user/index.cfm]

[DL99] United States Department of Defense, Defense Link, News Release, NO.032-99, *Department of Defense Budget For FY 2000*, [http://www.defenselink.mil/news/Feb1999/b02011999_bt032-99.html], 1 February 1999.

[DoD91] DoD Directive 8320.1, *DoD Data Administration*, [http://web7.whs.osd.mil/pdf/d83201p.pdf], September 26, 1991.

[FDM01] DoD, *Functional Description of the Mission Space Resource Center*, [http://fdms.msiac.dmso.mil/]

[GEF01] University of California at Irvine, *Graph Editing Framework (GEF)*, [http://www.ics.uci.edu/pub/arch/gef]

[JSR31] Java Specification Request 31, "XML Data Binding Specification", [http://jcp.org/jsr/detail/031.jsp]. August 1999.

[Leff00] Leffingwell, D., Widrig, D., *Managing Software Requirements A Unified Approach*, Addison-Wesley, 2000.

[KA95] Khoshafian, S., Abnous, R., *Object Orientation*, John Wiley and Sons, Inc., New York, NY, 1995.

[KM98] Kahng, J., McLeod D., "Dynamic Classificational Ontologies: Mediation of Information Sharing in Cooperative Federated Database Systems", Cooperative Information Systems, Trends and Directions, Academic Press, 1998.

[Lytt00] Lyttle, B., Ehrhardt, T., *Interconnectivity via a Consolidated Type Hierarchy and XML*, Master's Thesis, Naval Postgraduate School, Monterey, CA, 2000.

[OECD01] Organization for Economic Co-operation and Development, *Gross Domestic Product Table*, [http://www.oecd.org/std/gdp.htm], Jul 2001.

[Pit97] Pitoura, E., "Providing Database Inter-operability through Object-Oriented Language Constructs", Journal of Systems Integration, Volume 7, No. 2, August 1997, pp. 99-126.

[Pug01] Pugh, R., *Methods For Determining Object Correspondence During System Integration*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 2001.

[SB01] SeeBeyond Technology Corp., *SeeBeyond Taking the Challenges out of eBusiness Integration, Making eBusiness a Reality*, Marketing Overview, [http://www.seebeyond.com/aboutus/], Jul 2001.

[WCS00] Walsh, A., Couch, J., Steinberg, D., *Java 2 Bible*, IDG Books Worldwide, Inc., Foster City, CA, 2000.

[Whatis] Whatis?com web site, [http://whatis.techtarget.com/definition/0,,sid9_gci212702,00.html], Last updated on: Nov 24, 2000

[Wie93] Wiederhold, G., "Intelligent Integration of Information", ACM-SIGMOD 93, Washington, DC, May 1993, pp. 434-437.

[YBGL01] Young, P., Berzins, V., Ge, J., Luqi, “Using an Object Oriented Model for Resolving Representational Differences between Heterogeneous Systems”, paper submitted to the 17th ACM Symposium on Applied Computing, Madrid, Spain, 10-14 March 2002.

[Young02] Young, P., *Integration of Heterogeneous Software Systems Through Computer-Aided Resolution of Data Representation Differences*, Ph.D. Dissertation, Naval Postgraduate School, Monterey, CA, Unpublished.

[Zeus01] Email reply to question between Brett McLaughlin, Lutris Technologies, and the authors, 20 August 2001.

THIS PAGE INTENTIONALLY LEFT BLANK

BIBLIOGRAPHY

[Hun00] Hunter, D., and others, *Beginning XML*, Wrox Press Ltd, August 200.

[Kay00] Kay, M., *XSLT Programmer's Reference*, Wrox Press Ltd, July 2000.

[McL00] McLaughlin, B., *Data binding from XML to Java applications*. Four part-series. IBM developerWorks, Part 1 [http://www-106.ibm.com/developerworks/library/data-binding1/index.html?dwzone=xml], Part 2 [http://www-106.ibm.com/developerworks/library/data-binding2/index.html?dwzone=xml], Part 3 [http://www-106.ibm.com/developerworks/library/data-binding3/index.html?dwzone=xml], Part 4 [http://www-106.ibm.com/developerworks/library/x-databind4/?dwzone=xml], July-October 2000.

[Sun99] Reinhold, M., *An XML Data-Binding Facility for the Javaä Platform*, Sun Microsystems, Inc., [http://java.sun.com/xml/jaxp-1.0.1/docs/bind.pdf]. 30 July 1999.

[Sun01] Reinhold, M., *The Javaä Architecture for XML Binding (JAXB), Working-Draft Specification*, Sun Microsystems, Inc., [ftp://ftp.java.sun.com/pub/xml/987dfjaxb10ea3ds/jaxb-0_21-wd-spec.pdf]. 30 May 2001.

[W3C] World Wide Web Consortium, *web site*, [http://www.w3c.org].

[W3Csch] World Wide Web Consortium, *XML Schema web site*, [http://www.w3c.org/XML/Schema].

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road, Suite 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, CA 93943-5101

3. Marine Corps Representative 1
   Naval Postgraduate School
   Monterey, California
   debarber@nps.navy.mil

4. Director, Training and Education, MCCDC, Code C46 1
   Quantico, Virginia
   webmaster@tecom.usmc.mil

5. Director, Marine Corps Research Center, MCCDC, Code C40RC 1
   Quantico, Virginia
   ramkeyce@tecom.usmc.mil
   strongka@tecom.usmc.mil
   sanftlebenka@tecom.usmc.mil

6. Marine Corps Tactical Systems Support Activity (Attn: Operations Officer) 1
   Camp Pendleton, California
   doranfv@mctssa.usmc.mil
   palanaj@mctssa.usmc.mil

7. Professor Luqi 1
   Naval Postgraduate School
   Monterey, California
   LUQI@nps.navy.mil

8. Professor Valdis Berzins 1
   Naval Postgraduate School
   Monterey, California
   berzinzs@nps.navy.mil

9. CAPT Paul Young 1
Naval Postgraduate School
Monterey, California
peyoung@nps.navy.mil

10. Major Brent P. Christie 1
149 LeHavre Drive
Cheektowaga, NY 14227
brent@christiez.com